

ISSN 0933-9434 | Juli/August 1989 | 3. Jg./Heft 4
ÖS 150,- | DM 19,80 | sfr 19,80

CD-ROM läutet Gigabyte-Zeitalter ein

MS OS/2:
Monitore und I/O-Subsysteme

SAA-Serie:
Eine Help-Engine für SAA-Applikationen

Jetzt neu:
QuickC 2.0

Das Multiple Document Interface unter Windows



Liebe Leser,

wenn Sie diese Ausgabe des *Microsoft System Journals* aufmerksam durchblättern, werden Ihnen sicherlich einige Veränderungen auffallen. So kümmern wir uns nicht mehr nur allein um die Software, sondern berichten auch über Hardware - sofern diese im direkten Zusammenhang mit der Softwareentwicklung steht.

Das ist allerdings nur der Anfang der Umgestaltung, die mit der nächsten Ausgabe - mit der wir übrigens das zweijährige Bestehen des *Microsoft System Journals* feiern - erst richtig losgeht.

Dann nämlich werden wir Sie mit einem neuen gefälligeren Layout und einem Umfang von 140 bis 160 Seiten überraschen. Außerdem wird in jedem Heft nicht nur die Diskette mit den Programmlistings, sondern auch ein C-Programmierkurs enthalten sein. Sollten Sie die Programmdiskette bisher abonniert haben, wird Ihnen das Restguthaben auf den Abo-Preis für die Zeitschrift angerechnet.

Und damit das *Microsoft System Journal* nicht nur eine Zeitschrift für Insider bleibt, wird der Vogel Verlag den Vertrieb auch über Kiosk und Fachbuchhandel vornehmen. Natürlich können Sie das *Microsoft System Journal* auch weiterhin abonnieren bzw. Ihren Kollegen und Bekannten zum Abonnement empfehlen. Daß Sie das bisher schon ausgiebig praktiziert haben, merken wir an der ständig wachsenden Leserschaft.

Dafür ein herzliches Dankeschön!

Christian Wedell

Geschäftsführer der Microsoft GmbH

Ausgabe Juli/August 1989





Der kommende Standard zur Handhabung von Dokumentfenstern:

Die Mehrfachdokumentenschnittstelle (MDI)

Die Mehrfachdokumentenschnittstelle (im englischen Multiple Document Interface, im folgenden MDI genannt) ist eine Benutzerschnittstelle für Microsoft Windows und den OS/2-Presentation Manager (im folgenden PM genannt), die die Darstellung mehrerer Tochterfenster innerhalb des Applikationsfensters ermöglicht. Jedes dieser kleineren Tochterfenster kann zur Anzeige unterschiedlicher Daten oder zur unterschiedlichen Anzeige gemeinsamer Daten verwendet werden.

Dieser Artikel beschreibt MDI, wobei sowohl auf die Benutzerschnittstelle (*Bild 1*) als auch auf die Programmierung dieses Standards eingegangen wird. In diesem Zusammenhang wird eine Windows-Bibliothek mit Namen MDI.LIB beschrieben, die die einfache Einbindung von MDI in Ihre eigene Applikation ermöglicht. Die Anwendung der Bibliothek wird anhand einer einfachen Applikation (COLORS.EXE) gezeigt. Schließlich wird der MDI-Standard der »Systems Application Architecture« (SAA) und der einheitlichen Benutzeroberfläche »Common User Access« (CUA) von IBM gegenübergestellt und verglichen.

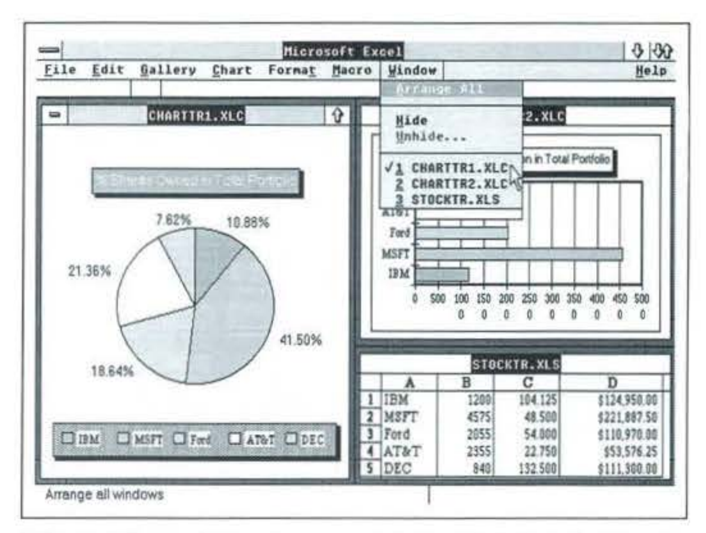


Bild 1: *Microsoft-Excel verwendet die Mehrfach-Dokumenten-Schnittstelle.*

Hintergrund und Motivation

Windows- und PM-Entwickler sind schon seit langer Zeit fasziniert von Applikationen, die Fenster innerhalb anderer Fenster darstellen. Diese Begeisterung beruht einerseits auf den physikalischen Fähigkeiten der Hardware und wird auch durch andere Fenstersysteme beeinflußt. MDI kann deshalb als weitere Verbesserung dieser Idee angesehen werden, indem versucht wurde, unter Windows eine Umgebung mit gemeinsamen Menüs zu erzeugen. Nachdem die Arbeit an diesem Standard beendet ist und erhebliche Veränderungen in die Windows-Oberfläche gebracht hat, wird MDI nun auf den OS/2-PM übertragen.

In gewissem Maße erfüllt MDI nicht nur die prinzipiellen Anforderungen der Entwickler, sondern andererseits wird dadurch auch die Funktionsweise von Mehrfachfenstern auf IBM-kompatiblen PCs festgelegt. Zusätzlich erleichtert die Spezifikation die Entwicklung einer neuen Generation von Applikationen mit ähnlicher Schnittstelle auf Apple Macintosh, Windows und OS/2-PM.

Von Beginn an wurde die MDI-Spezifikation mehrfach verbessert und erweitert (vorwiegend von Microsoft), um sie mit CUA in Einklang zu bringen. Das Ergebnis war die formale Definition von MDI im Handbuch des Microsoft Windows Software Development Kit (SDK), gefolgt von der Verwendung in Microsoft-Excel - der ersten Applikation mit MDI-Oberfläche.

Dem Vorbild Microsoft-Excel folgend haben viele Software-Entwickler versucht, mehrfache Tochterfenster in ihre Applikationen einzubauen, aber unglücklicherweise gelang es nur wenigen, die Originalspezifikation von MDI vollständig zu implementieren. Dies kommt daher, daß es ziemlich schwierig ist, MDI korrekt zu implementieren, da dies ein detailliertes und tiefgreifendes Verständnis der zugrundeliegenden Systemumgebung erfordert. Weiterhin benötigt die Implementierung von MDI bei Windows eine Reihe raffinierter Tricks, die an schlechten Programmierstil zumindestens erinnern.

Ungeachtet dieser Schwierigkeiten wurde die Akzeptanz von MDI durch das neue (nach meiner Auffassung nicht sehr gut durchdachte) OS/2-Dateisystem (siehe *Bild 2*) weiter gefestigt. Verbunden mit dem neuen und auch konsistenteren PM-Programmiermodell wird dies wahrscheinlich seine Anziehungskraft sowohl auf Entwickler als auch auf Software-Vertreiber vergrößern.

Definition und Spezifikation

Von Anfang an muß klargestellt werden, daß MDI die Darstellung einer Benutzerschnittstelle ist, was bedeutet, daß MDI nicht eine Menge fester Implementierungsregeln beinhaltet, sondern eine Sammlung von Richtlinien darstellt. Obwohl sich viele Entwickler bei der Implementierung von MDI an diese Richtlinien halten, haben nur wenige MDI auf die gleiche Weise implementiert.

Das erste, was man bei MDI verstanden haben muß, besteht darin, daß es ein oberstes Fenster gibt, das *Hauptfenster* der Applikation. Dieses Fenster ist immer in der Größe veränderbar, besitzt eine Titelzeile im Kopf, ein Steuermenüfeld und Sinnbildfelder zum Verkleinern und Vergrößern des Fensters. Meistens enthält die Titelleiste des Hauptfensters nur den Namen der Applikation (mehr dazu weiter unten). Wie bei den meisten anderen Fenstern auch, kann das Steuerungsmenü mit den Abkürzungstasten auf der folgenden Seite bedient werden.

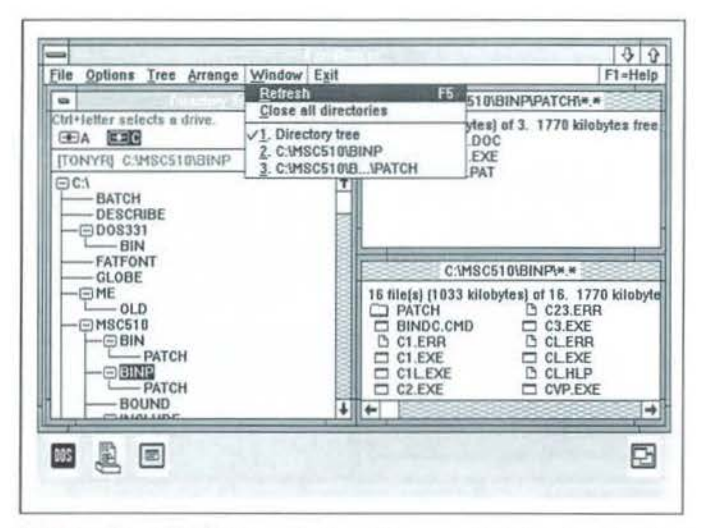


Bild 2: *Das OS/2-Presentation-Manager-Dateisystem basiert auf MDI.*

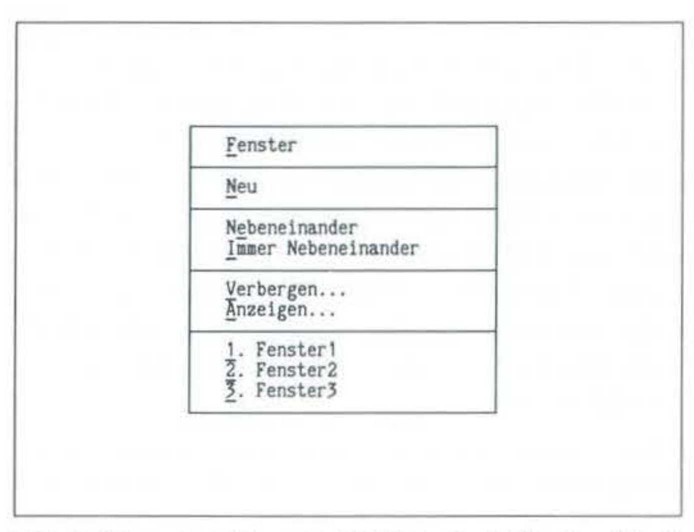


Bild 3: *Ein aufgeschlagenes MDI-Standard-»Fenster«-Menü.*

Wiederherstellen	Alt F5
Bewegen	Alt F7
Größe ändern	Alt F8
Sinnbild	Alt F9
Vollbild	Alt F10
Schließen	Alt F4

Dieses Hauptfenster zeigt auch die Menüleisten an, die den einzelnen vorhandenen Tochterfenstern zugeordnet sind. Die Menüleiste verändert sich je nachdem, welches der Tochterfenster gerade aktiv ist. Wechselt beispielsweise der Benutzer von einem Tochterfenster, welches ein Diagramm darstellt, zu einem anderen, welches ein Rechenblatt der Tabellenkalkulation beinhaltet, verändert sich im Hauptfenster die Menüleiste, um nun die Menübefehle des neuen aktiven Fensters anzuzeigen. Einige Menüeinträge können gleichbleiben, unabhängig davon, welches Fenster gerade angewählt wurde. So haben beispielsweise Datei- oder Formatierbefehle die gleiche Bedeutung in Fenstern mit unterschiedlichem Kontext.

Weiterhin enthält die Menüleiste des Hauptfensters ein weiteres Menü zum Verwalten der Tochterfenster. Die Befehle in diesem Menü sind unabhängig davon, welches Fenster gerade ausgewählt ist. Somit kann der Benutzer mit diesem Menü die Anordnung der Tochterfenster im Hauptfenster einstellen. Im ersten Teil des Menüs befinden sich Befehle, welche Größe, Position und Sichtbarkeit des Tochterfensters behandeln. Im zweiten Teil befindet sich eine Liste aller (auch als Sinnbild) sichtbaren Fenster. Meistens sieht dieses Menü ungefähr so aus wie in *Bild 3*.

Der Befehl »Neu« ermöglicht dem Benutzer, eine weitere Abbildung des augenblicklich ausgewählten Dokuments anzuzeigen. Hierzu wird ein neues Tochterfenster erzeugt, welches ebenfalls das ausgewählte Dokument enthält. Obwohl dies nicht immer sinnvoll ist, ist es dann nützlich, wenn ein Benutzer verschiedene Ansichten eines Dokuments gleichzeitig betrachten möchte. Es könnte bespielsweise dazu verwendet werden, verschiedene Vergrößerungen einer Zeichnung in einem Grafikprogramm anzuzeigen: Ein Fenster enthält die gesamte Zeichnung, ein anderes einen vergrößerten Ausschnitt der Zeichnung.

Obwohl der vorhandene Bildschirm von der Größe meist ziemlich beschränkt ist, bieten die meisten MDI-Implementierungen einige Mechanismen zur Anordnung der verschiedenen Tochterfenster an. In der Implementierung in diesem Artikel folgen dem »Neu«-Befehl zwei weitere Befehle, die die Anordnung der Tochterfenster erleichtern helfen.

Der Befehl »Nebeneinander« legt alle sichtbaren Tochterfenster in den Zeichenbereich des Hauptfensters. Obwohl viele gute Algorithmen zur Anordnung denkbar sind, verfahren die meisten Implementierungen derart, daß sie dem augenblicklich angewählten Tochterfenster eine höhere Prioriät zuordnen und ihm den meisten Platz zuweisen. Zu beachten ist, daß der »Nebeneinander«-Befehl wie bei den meisten Implementierungen nur zum Zeitpunkt seines Aufrufens die Fenster nebeneinander legt. Spätere Verschiebungen der Tochterfenster zerstören die Anordnung wieder.

Der Befehl »Immer Nebeneinander« ist eine Erweiterung des »Nebeneinander«-Befehls. Er bewirkt, daß die Tochterfenster auch in Zukunft nebeneinander stehen bleiben. Wird die Größe eines Tochterfensters verändert, verändern sich die Größen der anderen Fenster so, daß sie wieder nebeneinander liegen. Irgendwelche Änderungen beim Mutterfenster oder eines der Tochterfenster bewirken, daß die Tochterfenster sich wieder neu anordnen (wie dies beim früheren Windows mit Version 1.03 erfolgte). Obwohl diese Option nicht bei anderen wichtigen Windows- oder PM-Applikationen verwendet worden ist, hat sie eine Reihe von Vorteilen, die ihre Verwendung ernsthaft rechtfertigt.

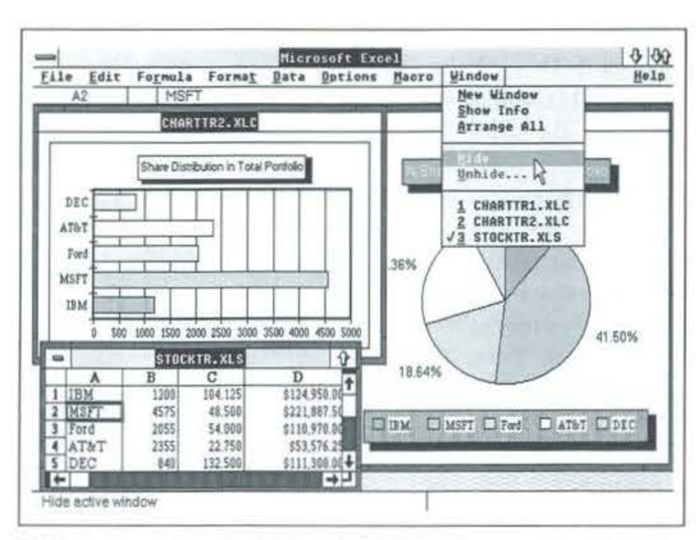


Bild 4: *Der Standard-Menübefehl »Verbergen« ist Teil des MDI-»Fenster«-Menüs.*

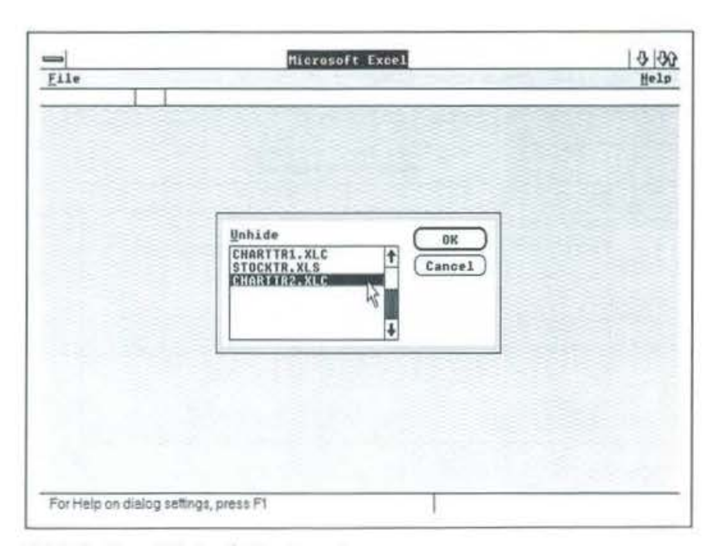


Bild 5: *Das Dialogfeld »Anzeigen«.*

Professionelle Anwender arbeiten oft mit mehreren gleichzeitig geöffneten Dokumenten, was zu einer unübersichtlichen Benutzeroberfläche führt. Deshalb folgen den »Nebeneinander«-Befehlen zwei Befehle, die das Verbergen oder das Wiederanzeigen eines oder mehrerer Dokumentfenster ermöglichen. Der »Verbergen«-Befehl bewirkt, daß das zuletzt vom Benutzer selektierte Tochterfenster verborgen wird (siehe *Bild 4*). Eine weitverbreitete Variante des Befehls ermöglicht dem Benutzer gleichzeitig mehrere Fenster zu verbergen, indem diese in einem Verzeichnisfeld innerhalb eines Dialogfelds gleichzeitig ausgewählt werden. Dadurch kann der Benutzer mit einem Handgriff einen Großteil der Darstellung verschwinden lassen.

Wenn Fenster verborgen sind, existieren sie weiterhin, es kann auf sie aber nicht zugegriffen werden. Mit Hilfe des Befehls »Anzeigen« kann der Benutzer über das Verzeichnis der verborgenen Fenster eines oder mehrerer davon wieder sichtbar machen. Die Fenster werden an die alte Position in ihrer ursprünglichen Größe wieder angeordnet oder, wenn die Option »Immer Nebeneinander« eingeschaltet ist, in die Fensteranordnung hineingeschoben und diese entsprechend korrigiert.

Die letzte Gruppe von Einträgen im Menü »Fenster« besteht aus einer Liste aller zur Zeit abgebildeten Fenster. Die Fenster sind mit ihren Titelnamen aufgeführt, wobei jedem Namen eine Ziffer vorangeht, die als Abkürzungstaste verwendet wird. Dies ermöglicht einen schnellen und einheitlichen Zugriff auf jedes Tochterfenster über die Tastatur, unabhängig vom gegenwärtigen Namen. Wenn eines der Fenster in der Liste aktiv ist, wird dies durch eine Markierung vor dem Namen angezeigt.

In einigen Applikationen können bestimmte Befehle nur dem Hauptfenster zugeordnet sein. In diesem Fall wird der Titelname des Hauptfensters am Beginn der Liste aufgeführt. Dies ermöglicht dem Benutzer einen einfachen Zugriff auf die Befehle des Hauptfensters. Applikationen, die diese Variante nicht benötigen, können den Namen des Hauptfensters in der Fensterliste weglassen.

Tochterfenster

Nachdem der Begriff des Hauptfensters klar geworden ist, muß man sich mit den Tochterfenstern auseinandersetzen, die durch MDI verknüpft sind. Wie das Hauptfenster, so ist jedes der Tochterfenster in der Größe veränderbar und enthält eine Titelleiste. Diese enthält normalerweise den Namen des Dokuments, das über das Fenster bearbeitet wird. Wird ein einzelnes Dokument in mehr als einem Tochterfenster angezeigt, wird an den Dokumentnamen eine Durchzählnummer angehängt, etwa:

```
CHART.XLC:1
CHART.XLC:2
CHART.XLC:3
```

Nur *ein* Tochterfenster kann zu einem bestimmten Zeitpunkt aktiv sein und dieses unterscheidet sich von den anderen Tochterfenstern durch die abweichende Farbe in der Titelleiste, genauso wie sich gewöhnlich das aktive Hauptfenster von anderen Applikations-Hauptfenstern unterscheidet. Zu beachten ist, daß das Hauptfenster aktiv bleibt, wenn eines der Dokumentfenster aktiviert wird. In gewisser Hinsicht ist dies ein sichtbarer Widerspruch, da der Eingabefokus gleichzeitig zwei Fenstern zugewiesen zu sein scheint. Er sagt noch nichts darüber aus, welche Purzelbäume Sie beim Programmieren schlagen müssen, um dies zu realisieren.

Das aktive MDI-Tochterfenster enthält auch ein Steuermenü. Obwohl es so ähnlich wie das des Mutterfensters benutzt wird, erfolgt der Aufruf über die Tastenkombination [Alt][-] (ich bin mir nicht ganz klar, welcher tieferliegende Grund sich dahinter verbirgt). Die Befehle im

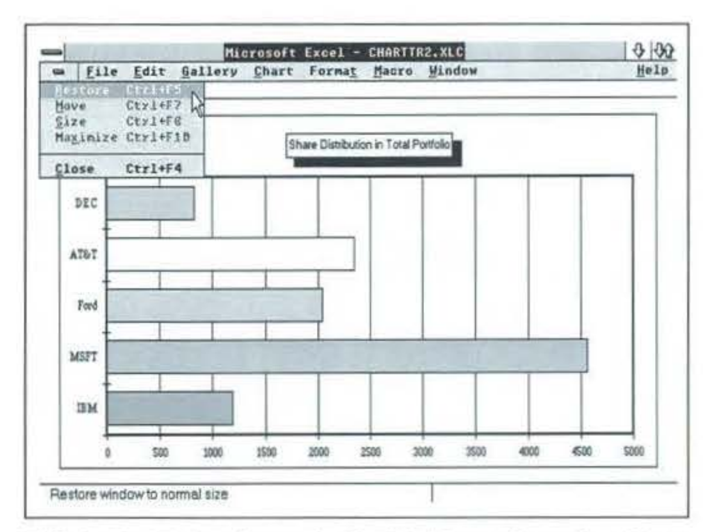


Bild 6: *Ein Tochterfenster in Vollbild-Darstellung. Der Name des Tochterfensters wurde in die Titelleiste von Microsoft Excel integriert.*

Steuermenü des Tochterfensters entsprechen denen des Hauptfensters, außer daß bei den Abkürzungstasten die [Alt]-Taste durch die [Strg]-Taste ersetzt wurde. Als Ergebnis erhält man ein Steuermenü, das die folgenden Einträge besitzt:

Wiederherstellen	[Strg][F5]
Bewegen	[Strg][F7]
Größe ändern	[Strg][F8]
Sinnbild	[Strg][F9]
Vollbild	[Strg][F10]
Schließen	[Strg][F4]

Es ist Aufgabe der Applikation, die ungeeigneten Einträge zu sperren oder grau anzuzeigen. Bei den meisten Implementierungen wird nur der »Sinnbild«-Befehl gesperrt.

Die Befehle »Bewegen« und »Größe ändern«, angesprochen über [Strg][F7] und [Strg][F8], erlauben die Position und Größe des Tochterfensters zu ändern. Die Funktionalität entspricht den entsprechenden Befehlen der Hauptfenster, aber ihre Ausdehnung ist bei Tochterfenstern auf den Fensterinhalt des Hauptfensters beschränkt. Wie bei den meisten Befehlen kann das Bewegen oder das Ändern der Größe eines Tochterfensters auch mit der Maus durchgeführt werden.

Ein interessantes Detail ist, wie der Fensterrahmen dargestellt wird, wenn das Fenster bewegt wird. Obwohl die Mausaktivitäten auf den Fensterinhalt des Hauptfensters beschränkt bleiben, kann der Bewegungsrahmen des Fensters auch außerhalb der Grenzen des Mutterfensters betrachtet werden. Bei der PM-Implementierung von MDI wird auch dieser Rahmen vom System am Rande des Hauptfensterinhalts abgeschnitten.

Der Befehl »Sinnbild« ([Strg][F9]), nur selten bei Windows implementiert, verkleinert das Tochterfenster zu einem Sinnbild, das innerhalb des Hauptfensterinhalts angezeigt wird. Das erhaltene Sinnbild kann ausgewählt, innerhalb des Hauptfensters bewegt und wieder zur ursprünglichen Größe und Position umgewandelt werden. In Übereinstimmung mit allen sichtbaren Tochterfenstern des MDI-Hauptfensters kann das Sinnbild verborgen oder über die Liste im »Fenster«-Menü ausgewählt werden. Zu beachten ist, daß während dieses Prozesses das Sinnbild innerhalb des Fensterinhalts des Hauptfensters stehen muß und nicht außerhalb davon auf dem Bildschirm bewegt werden darf. Beim OS/2-PM ist dies relativ einfach zu realisieren, bei Windows dagegen steigt dadurch weiter die Komplexität der MDI-Implementierung an und deshalb sind MDI-Implementierungen mit Tochterfenster-Sinnbilder ziemlich selten. Beim OS/2-PM ist dies dagegen erheblich leichter zu implementieren, und ich erwarte, daß mehr Applikationen in der MDI-Implementierung die Vorteile dieser Fähigkeit nutzen werden. Wenn Sie neugierig sind, wie dies bei OS/2 gemacht wird, sehen Sie sich die Beispiel-Applikation WITHIN im MS-OS/2-Software-Development-Kit (OS/2-SDK), Version 1.1 an.

Der Befehl »Vollbild« ([Strg][F10]) bewirkt, daß das MDI-Tochterfenster so vergrößert wird, daß es den gesamten Fensterinhalt des Hauptfensters ausfüllt (siehe *Bild 6*). Als Abkürzung kann auch die Maus verwendet werden und damit das Vollbild-Sinnbild angeklickt oder die Titelleiste doppelt anklickt werden.

Wenn der Fensterinhalt des Tochterfensters das Hauptfenster ausfüllt, kann man sich vorstellen, daß sich die Titelleiste des Tochterfensters unter die Menüleiste des Applikationsfensters »schiebt«. Dann geschehen zwei weitere Dinge: Das erste betrifft die Titelleiste des Hauptfensters. Ursprünglich enthält sie nur den Applikationsnamen. Aber wenn ein Tochterfenster als Vollbild dargestellt wird, wird die Titelleiste des Applikationsfenster so geändert, daß sie den Namen des aktuellen Dokuments enthält, wie es bei normalen Applikationen ohne MDI üblich ist. Zweitens, und dies ist vielleicht sogar die kompliziertere Änderung, wird das Steuermenü des Tochterfensters an den Beginn der Applikations-Menüleiste gesetzt. Dadurch ist weiterhin der Zugriff auf das Tochterfenster-Steuermenü möglich, um es zu schließen oder auf die ursprüngliche Größe zu verkleinern.

Wenn Sie denken, daß diese Dinge schon kompliziert genug sind, sollten Sie noch bedenken, was passiert, wenn ein neues MDI-Tochterfenster angelegt oder ein bereits existierendes ausgewählt wird, während ein anderes ein Vollbild ist. Die MDI-Spezifikation schreibt vor, daß bei der Auswahl eines anderen Fensters oder beim Neuanlegen dieses Fenster automatisch die Charakteristik des vorher selektierten besitzt. Dies beinhaltet auch, daß beim Neuanlegen eines Fensters während der Vollbild-Darstellung eines anderen auch das neue Fenster als Vollbild dargestellt wird. Ähnlich verhält es sich beim Schließen eines Vollbild-Fensters: Die MDI-Applikation wählt automatisch das

Taste	Funktion
[F6]	Wähle nächstes Unterfenster (im Uhrzeigersinn)
[Umsch] [F6]	Wähle vorheriges Unterfenster (im Gegenuhrzeigersinn)
[Strg] [F6]	Wähle nächstes Tochterfenster (vom ersten zum letzten)
[Umsch] [Strg] [F6]	Wähle vorheriges Tochterfenster (vom letzten zum ersten)

Tabelle 1: *Standard-MDI-Abkürzungstasten*

nächste verfügbare Tochterfenster aus und vergrößert es für Sie - ob Sie nun wollen oder nicht.

Mit dem »Wiederherstellen«-Befehl ([Strg] [F5]) wird, wie Sie erwarten dürften, das Fenster aus seiner Vollbild-Darstellung (oder seiner Sinnbild-Darstellung, sofern implementiert) in seine ursprüngliche Größe und Position zurückgewandelt, zusammen mit den anderen Fenstern. Auch das Steuermenü wandert wieder an seinen alten Platz.

Schließlich zerstört der »Schließen«-Befehl ([Strg] [F4]) das aktuell ausgewählte Tochterfenster. In Situationen, in denen das Tochterfenster eine von mehreren Ansichten eines gemeinsamen Dokuments darstellt, werden die Titelleisten der übrigbleibenden Fenster automatisch neu durchnumeriert, um die Veränderung anzuzeigen. Wenn das Tochterfenster das letzte Fenster ist, das auf das Dokument zugreift, wird üblicherweise ein Dialogfeld angezeigt, das nachfragt, ob Änderungen gesichert werden sollen. Wie bei allen Steuermenüs ist das doppelte Anklicken des Steuermenüfelds eine Abkürzung für die Auswahl des »Schließen«-Befehls.

Eine andere Sache, die bei all den beschriebenen Befehlen beachtet werden sollte, ist, daß diese sich nur auf das augenblicklich aktive Tochterfenster beziehen. Das bedeutet auch, daß nur dieses Fenster ein Steuermenü und Felder für Vollbild oder Sinnbild besitzt. Unglücklicherweise geht dies aus der ursprünglichen MDI-Spezifikation nicht ganz klar hervor. Als abschließendes Ergebnis muß jedes der Tochterfenster seine Darstellungsattribute ändern, wenn es den Eingabefokus erhält oder entzogen bekommt.

Als wäre dies noch nicht genug, enthält die MDI-Spezifikation auch noch eine Liste von Abkürzungstasten für den Wechsel zwischen verschiedenen Tochterfenstern (siehe *Tabelle 1*). Trotz der Tatsache, daß diese Tasten nicht im Steuermenü des Tochterfensters aufgeführt worden sind, sind sie der einzige Mechanismus für den Wechsel zwischen Tochterfenstern ohne Verwendung der Maus. Es bleibt den Benutzern überlassen (vorausgesetzt sie lesen die Dokumentation), sich zu erinnern, daß es diese Tasten gibt und was sie bedeuten. Schließlich ist auch die Umsetzung dieser Abkürzungstasten eine weitere Aufgabe des bereits ziemlich überladenen MDI-Hauptfensters.

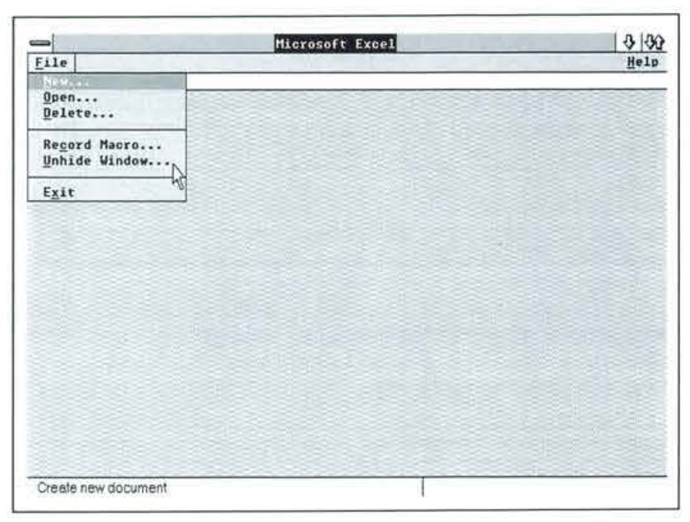


Bild 7: *Microsoft Excel, nachdem alle Tochterfenster verborgen wurden. Die meisten Menüs sind verschwunden und der Menü-Befehl »Anzeigen« ist jetzt Bestandteil des »Datei«-Menüs.*

Entwurfs-Aspekte

Bevor wir auf die eigentlichen Programmier-Aspekte eingehen, die aus der MDI-Implementierung resultieren, scheint es angebracht, die Entwurfs-Aspekte zu diskutieren, die auftauchen, wenn man sich mit der Spezifikation beschäftigt. Der vielleicht vordergründigste Aspekt ist die zusätzliche Komplexität, die mit der Verwendung von MDI einhergeht. Wenn Sie nicht inzwischen ein eigenes Konzept besitzen, werden Sie viel Zeit benötigen, bis MDI implementiert ist und korrekt funktioniert. Vom Entwurfsstandpunkt her fordert MDI, daß jedes Tochterfenster in seinem Aufbau objektorientiert ist (also seine eigenen Daten unabhängig verwaltet), dennoch kann es Zugriffe auf gemeinsame Daten geben, wenn mehrere Ansichten eines Objekts stattfinden. Weiterhin bewirkt der Standard einige Auswirkungen auf die Performance, da er zusätzliche Verwaltungen und eine weitere Hierarchie-Ebene in dem System einführt.

Es liegt auch auf der Hand, als Alternative mehrere Instanzen von eng verbundenen Applikationen mit einer MDI-Implementierung zu vergleichen. Positiv ist, daß eine Gruppe von unabhängigen Applikationen oft leichter zu entwerfen, implementieren und zu testen ist, insbesondere wenn die Umgebung die Aspekte der Datenverwaltung überwacht. Dieser Ansatz ist insbesondere dann erfolgreich, wenn nicht Windows-spezifische Applikationen verwendet werden oder der Einfluß jener auf die Umgebung unter günstigsten Umständen gering ist.

Negativ ist, daß sich ein unübersichtlicher Bildschirminhalt ergibt, sich keine Konsistenz mit der Benutzeroberfläche des Apple-Macintosh herstellen läßt und die Schwierigkeit, eine gut integrierte Interprozeß-Kommunikation

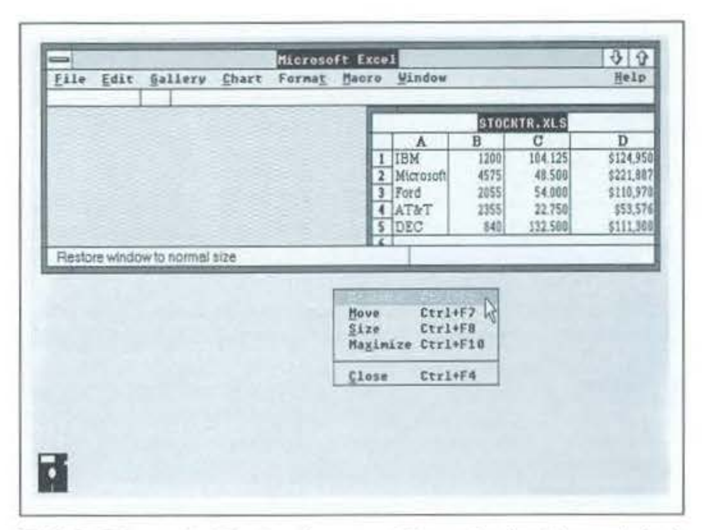


Bild 8: *Wenn ein Tochterfenster nicht mehr im Fensterinhalt des Hauptfensters steht, kann dennoch über die Tastatur darauf zugegriffen werden. Allerdings treten einige merkwürdige Effekte auf. Beispielsweise erscheint das Steuermenü des Tochterfensters »einsam« ohne Bezug auf dem Bildschirm.*

zwischen unabhängigen Applikationen zu schaffen. Weiterhin erfordern viele Applikation die mehrfache Betrachtung eines einzelnen Dokuments und somit ist die MDI-Implementierung etwas, über das Sie einmal nachdenken sollten.

Ein weiteres lästiges Feld beim Entwickeln ist die Art, wie MDI mit Menüs umgeht. Obwohl Menüs ziemlich einfach zu strukturieren sind, wenn jedes Tochterfenster homogen in seiner Beschaffenheit ist und gleiche oder ähnliche Fähigkeiten gemeinsam nutzt, kann der Entwurf kompliziert werden, wenn Tochterfenster sehr heterogen sind oder sich auf verbundene Dokumente beziehen.

Der Menü-Entwurf ist weiterhin dann kompliziert, wenn das Applikationsfenster leer ist oder alle Tochterfenster verborgen wurden. Die meisten der normalen Menüeinträge werden dann ungültig und für den Benutzer uninteressant. Viele Applikationen, beispielsweise Microsoft-Excel (siehe *Bild 7*), reagieren in dieser Situation dadurch, daß das vorhandene Menü stark verkleinert wird, was die Komplexität der Menüverwaltung weiter erhöht.

Eines der schwierigsten Probleme beim MDI-Standard taucht auf, wenn die Größe des Applikationsfensters verändert wird. Obwohl es nicht möglich ist, ein Tochterfenster komplett aus dem Zeichenbereich des Hauptfensters zu schieben, ist dies schließlich doch möglich, indem die Größe des MDI-Hauptfensters verändert wird. Das Resultat ist ein Fenster (das vielleicht sogar das aktive ist), welches vollständig unsichtbar ist. Zwar kann das Fenster nicht mehr mit der Maus angesprochen werden, aber man kann es über die Abkürzungstasten erreichen, aber sichtbar ist es dadurch nicht geworden.

Ein interessantes visuelles Phänomen kann erzeugt werden, indem man [Strg][-] eingibt, während das aktive Tochterfenster außerhalb des Inhalts des Hauptfensters steht. Wie erwartet wird das Steuermenü des Tochterfensters aufgeschlagen, aber das Tochterfenster selbst bleibt unsichtbar. Als Resultat ergibt sich, daß das Steuermenü ohne Bezug auf dem Bildschirm erscheint und sich wie durch Magie selbst sichtbar gemacht hat (*Bild 8*). Mag sein, daß so etwas interessant aussieht, aber es ist sicherlich auch verwirrend für Benutzer!

Das API des MDI

Bis jetzt mögen Sie sich fragen, was an MDI eigentlich aufregend ist - insbesondere wenn man bedenkt, daß es im Grunde nichts anderes ist als die Verwaltung einer Menge von zusammenhängenden Tochterfenster. Aber es gibt einen guten Grund, MDI einzusetzen - nämlich die Definition und Implementierung einer Applikationsprogramm-Schnittstelle (im Original *Application Program Interface*, im folgenden als API bezeichnet), die MDI organisiert. Das Endergebnis des API ist eine kleine Bibliothek aus Objektmodulen (ungefähr 16 Kbyte Gesamtgröße), die die gesamte Arbeit übernimmt, MDI in Ihre Applikation zu integrieren. Und als Bestes von allem: Sie ist im Preis der Ausgabe dieses *Microsoft System Journals* eingeschlossen und kann auch auf Diskette bezogen werden!

Um die Integrationsarbeit des API in eine Applikation zu schaffen, griff ich auf die Hilfe eines Freundes, langjährigen Kollegen und Windows-Gurus zurück - Geoffrey Nicholls. Zusammen entwickelten wir ein API, das es erlaubt, komplexe MDI-Applikationen zu schreiben, als wären sie einzelne unabhängige Applikationen.

Wir erkannten von Beginn der Entwicklung an, daß ein solches API ziemlich unsauber würde und wir Dinge tun mußten, die wir niemals bei der konventionellen Windows-Programmierung tun würden. Weiterhin mußten wir versuchen, das API klein und einfach zu halten. Uns wurde auch klar, daß wir nicht die gesamte Spezifikation würden implementieren können, sondern nur einige der wichtigeren Features - den Rest wollen wir Ihnen überlassen. Schließlich wollten wir alles so objektorientiert wie möglich machen. Nach mehreren Fehlstarts und Neuprogrammierungen haben wir unser Ziel erreicht, vielleicht mit etwas übertriebener Verwendung von Besitzlisten und zusätzlichen Fensterdatenbereichen.

Das MDI-API wurde so entwickelt, daß es ein Analogon zum existierenden Windows-API darstellt. Aus früherer Erfahrung wußten wir, daß in großem Maße die Charakteristik eines Fensters durch *die* Funktion festgelegt ist, die die Ersatzverarbeitung der Fensternachrichten übernimmt. Das MDI-API versucht, diesen Unterbau zu ändern und jedem Fenster einen neuen und unterschiedlichen Satz von Charakteren zu geben. Das Endresultat ist eine kleine Anzahl Routinen mit vertrauten Parameterlisten, welche zusammen verwendet werden, um Ihrer Applikation die MDI-Charakteristik zu verleihen. Die Routinen sind in *Tabelle 2* aufgeführt.

`MdiMainCreateWindow( )`
Lege MDI-Hauptfenster an

`MdiMainDefWindowProc( )`
Ersatz-Fensterfunktion des MDI-Hauptfensters

`MdiChildCreateWindow( )`
Lege ein MDI-Tochterfenster an

`MdiChildDefWindowProc( )`
Ersatz-Fensterfunktion eines MDI-Tochterfensters

`MdiGetMessage( )`
Lese nächste MDI-Applikations-Nachricht

`MdiTranslateAccelerators( )`
Übersetze Abkürzungstasten einer MDI-Applikation

`MdiSetAccel( )`
Setze Abkürzungstasten-Tabelle für ein MDI-Dokument

`MdiGetMenu( )`
Lese Menübezug eines MDI-Tochterfensters

Tabelle 2: *MDI-Applikations-Programmierschnittstelle*

Nachrichtenfluß und Prozeßablauf

In den nächsten zwei Abschnitten wollen wir die internen Abläufe im MDI-API untersuchen. Zuerst beschreiben wir den allgemeinen Nachrichtenfluß und den Prozeßablauf im API. Dann erläutern wir jede der Eintrittsfunktionen und erklären einige der ausgetüftelten Verfahren, auf denen sie beruhen. Wenn Sie diese Abschnitte lesen, sollten Sie auf die MDI-Quelldateien zurückgreifen, die in diesem Artikel aufgeführt sind. Der Code ist ziemlich gut dokumentiert und Sie müßten ihn verstehen können, sofern Sie über ein gutes Hintergrundwissen bei der Windows-Programmierung verfügen.

Nun, vielleicht der effizienteste Weg das MDI-API zu lernen, besteht darin, sorgfältig das MDI-Nachrichtenfluß-diagramm im *Bild 9* zu studieren. Es verfolgt die Pfade jeder Nachricht, die von einer Applikation empfangen werden, die das API benutzt, wobei nur die uns interessierenden Ausschnitte betrachtet werden.

Die erste Eigenschaft, die zu beachten ist, betrifft die ziemlich normal aussehende Nachrichtenschleife mit der Entnahme, Übersetzung und Verteilung einer Nachricht am oberen Anfang des Diagramms. Dies erfolgt ähnlich wie in beliebigen anderen Windows-Applikationen, der einzige Unterschied besteht in der spezifischen Überprüfung menübezogener Nachrichten (dies erfolgt in der Funktion `MdiGetMessage`). Wenn eine solche Nachricht auftaucht, wird sie sofort an die entsprechende Fensterfunktion verteilt, um die verschiedenen Steuer- und Applikationsmenüs richtig aufzurufen.

Nach der Nachrichtenschleife wird jede Nachricht an die entsprechende Fensterfunktion gesandt. Soweit MDI betroffen ist, sind nur zwei Arten von Fenstern präsent: Ein Applikations- oder Hauptfenster und die Tochter- oder Dokumentfenster. Auf der linken Seite des Diagramms wird der Nachrichtenfluß für das Applikations-Hauptfenster gezeigt; auf der rechten Seite ist ein ähnlicher Ablauf für jedes der Dokumentfenster aufgeführt.

Verfolgt man die linke Seite (die des Hauptfensters) wird jede Nachricht in der Fensterfunktion des Hauptfensters verarbeitet und anschließend der MDI-Ersatzfensterfunktion übergeben. Der Rest der Liste beschreibt die Ereignisse in dieser Ersatzfunktion, die in den meisten Fällen dadurch beendet wird, daß die Nachricht zur Standardersatzfunktion `DefWindowProc` gesandt wird.

Wie Sie in dem Diagramm sehen können, ist die MDI-Ersatzfunktion hauptsächlich interessiert an Nachrichten zum Aktivieren und Initialisieren sowie befehlsspezifischen Nachrichten. Alle anderen Nachrichten werden unverändert an `DefWindowProc` übergeben. Von den ersteren werden einige abgefangen (etwa die Aktivierung eines bestimmten Tochterfensters); andere werden verarbeitet und direkt an das entsprechende Tochterfenster gesandt - unter Umgehung der Ersatzfensterfunktion.

Auf der rechten Seite (die der Dokumentfenster) des Diagramms befindet sich die Sequenz von Ereignissen, die beim Empfang von Nachrichten durch die MDI-Tochterfenster-Funktion auftreten. Wie auf der linken Seite ist der Ablauf in der Ersatzfunktion aufgeführt, der in den Fällen damit endet, daß die Nachricht an die Standard-Ersatzfunktion `DefWindowProc` weitergeleitet wird.

Wie die Hauptfenster-Ersatzfunktion ist auch die Ersatzfunktion der Dokumentfenster vorwiegend interessiert an Nachrichten zum Aktivieren und Initialisieren sowie befehlsspezifischen Nachrichten. Wichtig sind insbesondere die verschiedenen Systembefehle. Einige werden direkt behandelt und nicht zum System geschickt. In anderen Fällen, beispielsweise wenn ein Menü oder ein neues Dokumentfenster aktiviert wird, wird die Nachricht direkt an das Hauptfenster geschickt.

An bestimmten Stellen im Diagramm kann man Felder mit dickerem Rahmen sehen, welche die Aktivierung eines Dokument-Fensters beschreiben. *Bild 10* zeigt den gleichen Ablauf, in kleinere Schritte aufgeteilt.

Bei einer normalen Windows-Applikation erfolgt die Aktivierung eines Fensters ohne größeres Geschrei, aber bei einer MDI-Implementierung müssen eine Reihe von wichtigen Schritten durchgeführt werden. Zunächst muß die Farbe in der Titelleiste geändert werden. Obwohl Windows dies nur für das eine Fenster erlaubt, welches den Eingabefokus besitzt, entsteht der Eindruck, daß beim Aktivieren eines Tochterfensters sowohl das Hauptfenster als auch das Tochterfenster gleichzeitig aktiv sind. Dies erfolgt, indem manuell eine `WM_NCPAINT`-Nachricht an `DefWindowProc` des zu aktivierenden Fensters geschickt wird.

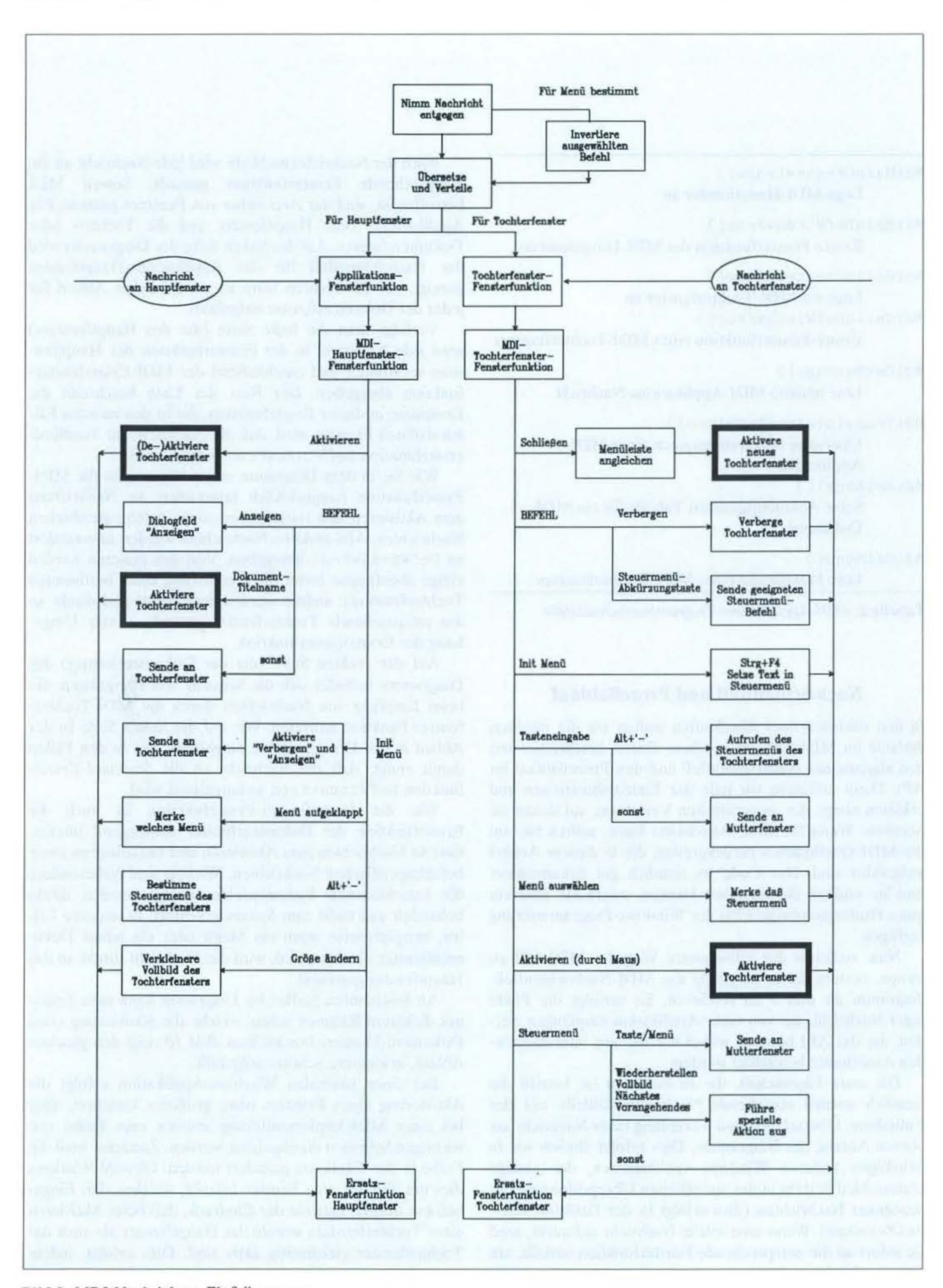


Bild 9: *MDI-Nachrichten-Flußdiagramm*

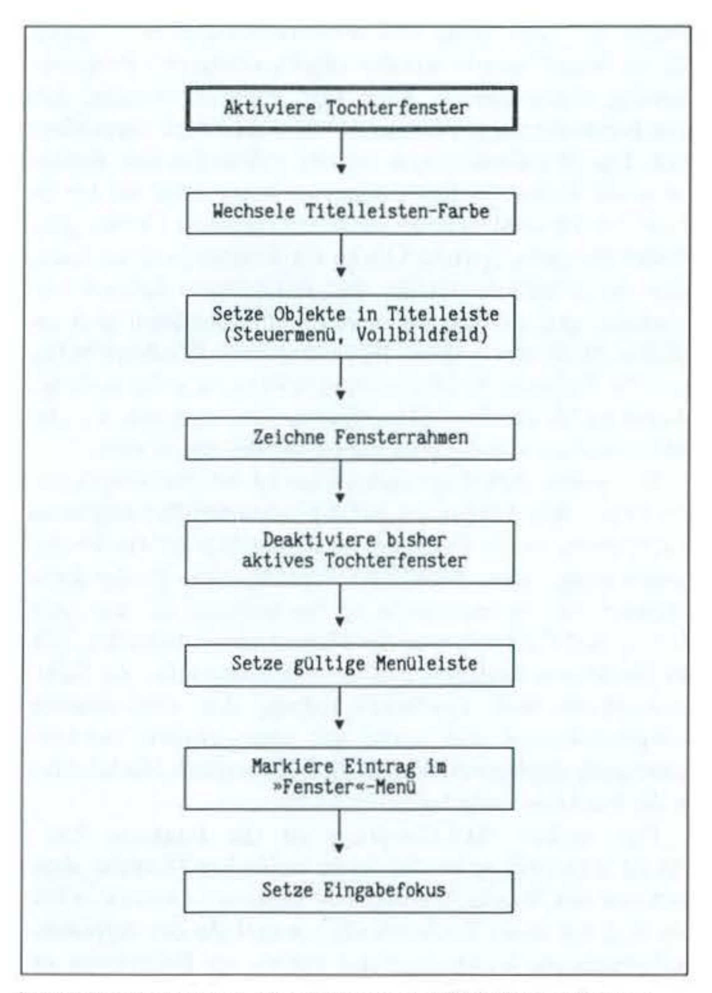


Bild 10: *Ablauf beim Aktivieren eines MDI-Tochterfensters.*

Weiterhin enthält laut Schnittstellendefinition nur das gegenwärtig aktive Dokumentfenster ein MDI-Steuermenü und Felder für Voll- und Sinnbild. Deshalb wird im zweiten Schritt der Fensterstil entsprechend geändert und anschließend das System aufgefordert, die Änderungen anzuzeigen.

Der nächste bedeutende Schritt beim Aktivieren eines Dokumentfensters besteht aus dem Setzen der Menüleiste im Hauptfenster. Jedes Tochterfenster besitzt seine eigene Menüleiste. Wenn es aktiviert wird, wird die augenblickliche Menüleiste entfernt und die neue eingefügt mit allen Attributen, die sie besitzt. Schließlich werden die Einträge im »Fenster«-Menü aktualisiert, um den aktuellen Status im Hauptfenster anzuzeigen. Erst dann enthält das Dokument abschließend den Eingabefokus.

Wie *Bild 10* enthalten die *Bilder 11* und *12* den Ablauf von Ereignissen, die sich ergeben, wenn ein Dokumentfenster als Vollbild dargestellt wird oder ein anderes Fenster ausgewählt wird, während das aktuelle ein Vollbild ist. Bei beiden Abläufen besteht die einzige technisch interessante Aktion aus dem automatischen Einfügen des MDI-Steuermenüs am Beginn der Applikations-Menüleiste.

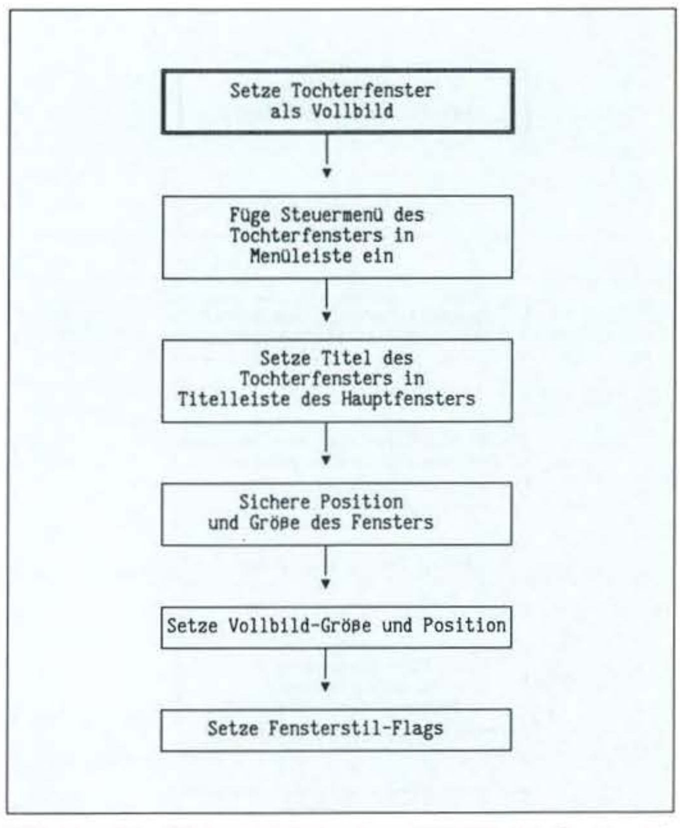


Bild 11: *Ablauf beim Abbilden eines MDI-Tochterfensters als Vollbild.*

Hierzu muß zunächst auf das MDI-Steuermenü im Windows-System zugegriffen werden, was so geschieht:

```
hBitmap=LoadBitmap(NULL,MAKEINTRESOURCE(OBM_CLOSE));
```

Das erhaltene Rasterbild enthält sowohl das Standard-Steuermenüfeld als auch das spezielle eines Tochterfensters. Nachdem die Dimensionen des Rasterbilds mit der Funktion `GetObject` ermittelt wurden, wird das Steuermenüfeld des Tochterfensters entnommen, um es in die Menüleiste der Applikation einzufügen. Der ganze Ablauf steht in der Funktion `MdiCreateChildSysBitmap` und war bisher nicht allgemein publiziert worden. Sie sollten die Neigung verspüren, sich anzuzusehen, was genau in dieser Funktion passiert!

Eintrittsfunktionen des API

Nachdem wir den Nachrichtenfluß und den Prozeßablauf von MDI diskutiert haben, wollen wir uns nun die Eintrittsfunktion ansehen, die das API zur Verfügung stellt. Programmiertechnisch gesprochen können Sie das MDI-Interface bei Ihren eigenen Applikationen ohne größere Probleme anwenden, wenn Sie die Bedeutung dieser Funktionen verstanden haben.

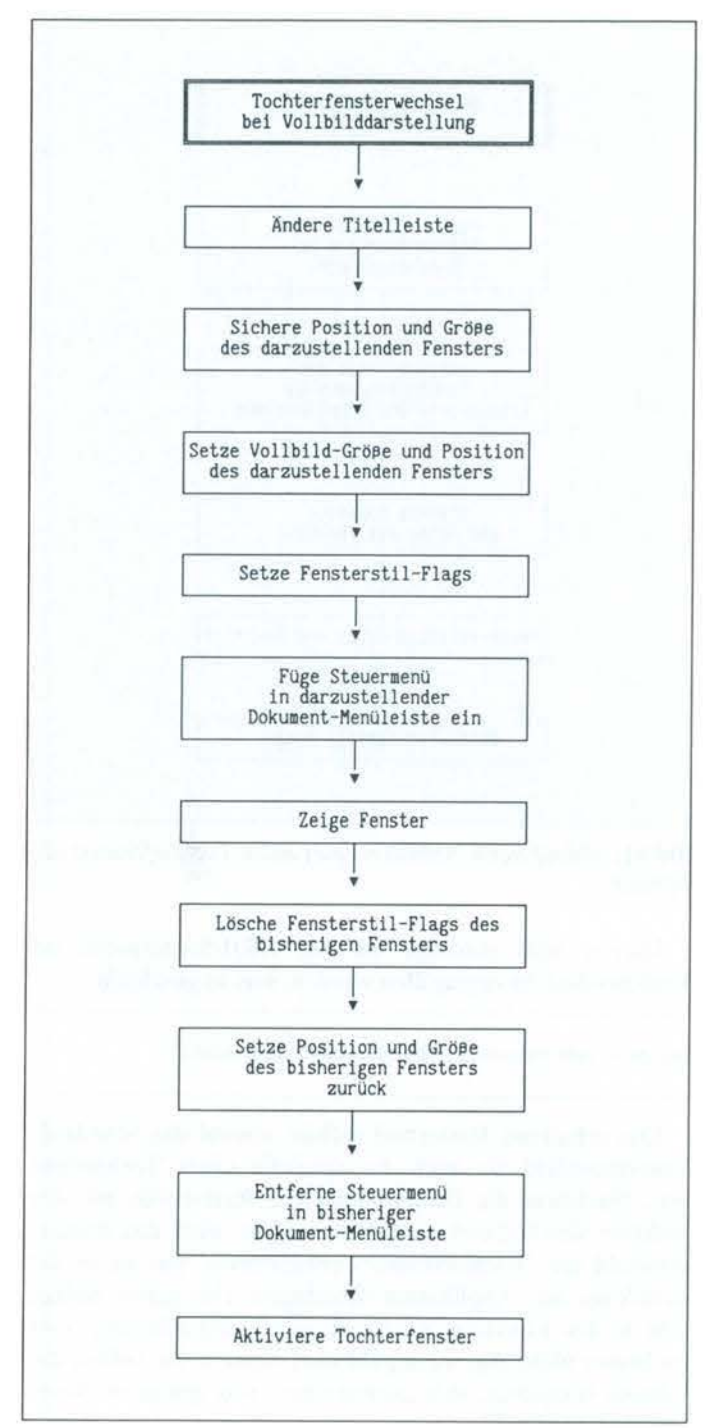


Bild 12: *Ablauf beim Wechseln zwischen MDI-Tochterfenstern in Vollbilddarstellung.*

Die erste Funktion ist `MdiMainCreateWindow`, die verantwortlich ist für das Erzeugen des Hauptfensters und alle zugeordneten MDI-Besitzlisten, welche für das Interface benötigt werden. Die eigentlichen Datenstrukturen, die vom API verwendet werden, werden in Besitzlisten (*property lists*) des Hauptfensters und der Dokumentfenster gehalten. Eine Besitzliste ist ein Versuch, jedem Fenster den Zugriff auf eine Reihe von Vorkommen-Daten zu geben (dieser Begriff wurde aus der objektorientierten Programmierung übernommen). Dies wird dadurch erreicht, daß dem Fensterbezug ein benannter Speicherblock zugeordnet wird. Das Windows-System erlaubt jedem Fenster, Besitzlisten anzulegen, zu bestimmen, zu lesen oder wieder zu zerstören. Obwohl es keine vordefinierte obere Grenze gibt, wieviel benannte Speicherblöcke ein Fenster besitzen kann, wird die maximale Größe der Besitzlisten dadurch beschränkt, daß sie wie die anderen Fensterdaten sich im lokalen Heap der »User«-Bibliothek von Windows befinden. Die Funktion `MdiMainCreateWindow` weist auch das »Fenster«-Menü dem Hauptfenster zu, dadurch ist das MDI-Interface ständig transparent für die Applikation.

Die zweite API-Eintrittsfunktion ist `MdiMainDefWindowProc`. Wie bereits im MDI-Nachrichtenflußdiagramm beschrieben, ist die Funktion verantwortlich für die Ersatzverarbeitung aller hauptfensterspezifischen Nachrichten. Genauer: Sie ist interessiert an Nachrichten, die die Aktivierung und Deaktivierung des Hauptfensters betreffen, sich auf die Menüs beziehen und an Nachrichten für die Sichtbarmachung und Größeneinstellung der zugeordneten Dokumentfenster. Sie wurde mit einer großen `switch`-Anweisung implementiert und gibt die meisten Nachrichten an die Funktion `DefWindowProc` weiter.

Der nächste API-Einsprung ist die Funktion `MdiChildCreateWindow`. Sie ist in vielfacher Hinsicht identisch mit der Standard-Windows-Funktion `CreateWindow` und legt ein neues Tochterfenster innerhalb des Applikationsfensters an. Im Hintergrund legt sie die Besitzlisten an, behält das »Fenster«-Menü und die Abkürzungstasten-Tabelle im Auge und aktiviert das Fenster so, wie es der augenblickliche Zustand der Applikation verlangt.

Wie beim Hauptfenster erhält jedes Dokumentfenster eine MDI-Funktion als Ersatzfunktion zum Verarbeiten der Nachrichten. Sie besitzt den Namen `MdiChildDefWindowProc` und ist verantwortlich für die Ersatzverarbeitung aller tochterfensterspezifischen Nachrichten, insbesondere jene zur Auswahl des Steuermenüs und zur Anlage, Aktivierung und Zerstörung des Fensters. Andere Nachrichten werden entweder direkt zum Hauptfenster gesandt oder der Funktion `DefWindowProc` übergeben.

Nach der Tochterfenster-Ersatzfunktion gibt es mit der Funktion `MdiGetMessage` einen Ersatz für die entsprechende Windows-Standardfunktion. Jene ist verantwortlich für die Entgegennahme aller Applikationsnachrichten aus dem Nachrichteneingangspuffer. Sie überprüft auch den Tastaturzugriff auf das Menü und aktiviert das korrekte Menü, wenn die Richtungstasten verwendet werden, um ein anderes Menü auszuwählen.

Als nächstes folgt die Funktion `MdiTranslateAccelerators`, welche jede Nachricht gemäß der aktuell gesetzten Abkürzungstastentabelle übersetzt. Obwohl die meisten Applikationen nur eine solche Tabelle besitzen, können MDI-Applikationen mehrere von ihnen besitzen -

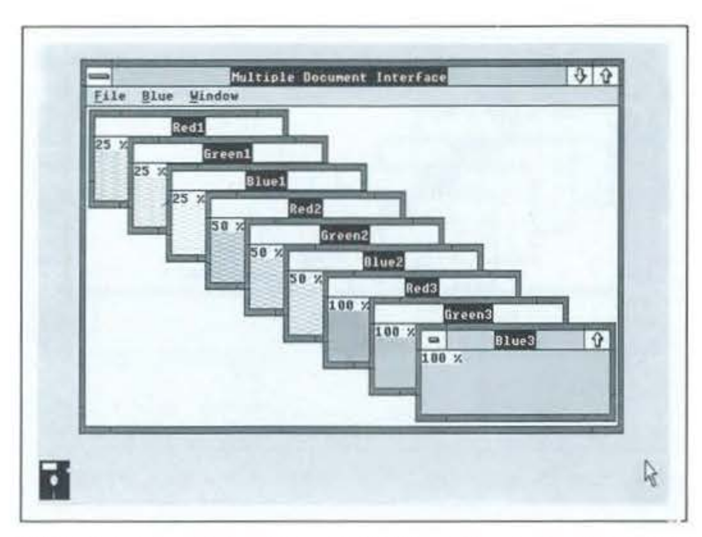


Bild 13a: *Das Programm COLORS kann rote, grüne und blaue Tochterfenster mit unterschiedlichem Grauwertanteil anlegen. Jedes neue Tochterfenster wird automatisch in einer kaskadenähnlichen Reihe positioniert.*

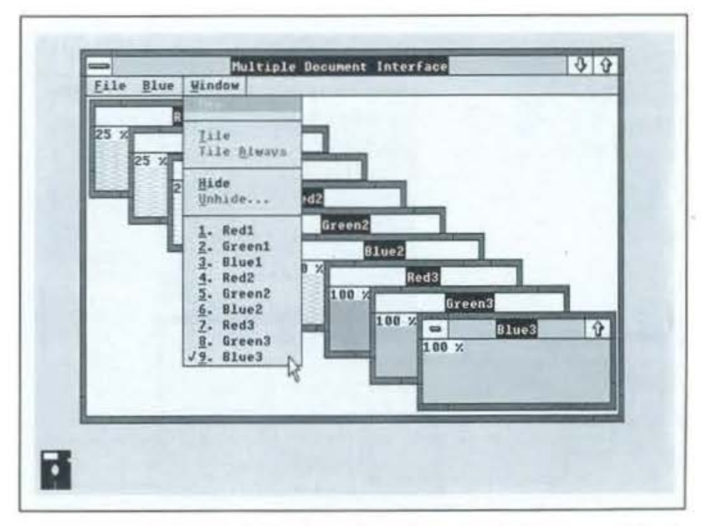


Bild 13b: *COLORS besitzt das Standard-»Fenster«-Menü, welches eine Liste aller angezeigten Fenster besitzt. Mit weiteren Befehlen können die Fenster verborgen und wieder angezeigt werden.*

eine für das Hauptfenster und je eine für einen Dokumentfenster-Typ. Die Funktion überprüft automatisch den Status der Applikation und verwendet die passende Abkürzungstasten-Tabelle.

Schließlich wurden zwei Hilfsfunktionen in das MDI-API aufgenommen, die als Makro implementiert wurden und `MdiGetMenu` beziehungsweise `MdiSetAccel` heißen. Die beiden Funktionen sind erforderlich, weil die meisten Applikationen eine Abkürzungstasten-Tabelle anlegen und auf die aktuelle Menüleiste zugreifen. Dies erfolgt durch das `MdiGetMenu`-Makro, welches den Menübezug der entsprechenden Dokument-Menüleiste ermittelt. Das Makro `MdiSetAccel` dagegen setzt eine Abkürzungstasten-Tabelle in der Besitzliste des angegebenen Dokumentfensters. Die Tabelle kann dann automatisch benutzt werden, wenn Nachrichten übersetzt und verteilt werden, ohne daß die Applikation eingreifen muß.

Zusammengefaßt repräsentieren diese acht Funktionen das gesamte MDI-API. Obwohl die Funktionen nicht einfach ungestraft verwendet werden können, sollten sie sich doch gut in die meisten denkbaren Applikationen integrieren lassen. Wenn sie vorsichtig angewandt werden, verbergen sie die meisten feinen Einzelheiten von MDI und ermöglichen Ihnen, sich auf die programmspezifischen Probleme zu konzentrieren und nicht ein neues Verfahren für die Verwaltung von Tochterfenstern erfinden zu müssen. Die einzige ziemlich unschöne Angelegenheit ist die Installation eines Systemeingriffs (*hook*) für die System-Nachrichten. Dieser Eingriff fängt die Richtungstasten innerhalb der Menüverwaltung ab. Ohne den Eingriff wäre es sehr schwierig, ein wirklich authentisches MDI-Tastaturinterface zu implementieren.

Erzeugen von MDI.LIB

Um die MDI-API-Bibliothek anzulegen, müssen zunächst die Dateien `MDI1.C`, `MDI2.C` und `MDI3.C` (siehe *Listings 2, 3* und *4*) vorhanden sein. Weiterhin wird die Datei `MDI.H` benötigt (*Listing 1*), die auch die Schnittstelle zum MDI-Anwenderprogramm darstellt. Zusätzlich zu diesen Quelldateien werden das Microsoft-Windows-SDK, Version 2.1 und der Microsoft-C-Compiler, Version 5.1 benötigt.

Die MAKE-Datei zum Erzeugen der Bibliothek (*Listing 5*) übersetzt jedes der Module im »medium«-Speichermodell und kombiniert sie in eine Objekt-Bibliothek mit Hilfe des LIB-Programms, welches dem C-Compiler beiliegt. Diese Bibliothek kann dann ohne Änderung für jede MDI-Applikation verwendet werden, die mit dem »medium«-Speichermodell übersetzt wird. Falls gewünscht, kann man die Übersetzungsflags in der MAKE-Datei ändern und sich dadurch eine entsprechende Bibliothek für das »small«-, »compact«- und »large«-Modell schaffen.

Verwendung des MDI-API

Ich werde im folgenden das Programm `COLORS.EXE` verwenden, um zu zeigen, wie das MDI-API als Erweiterung einer einfachen Applikation verwendet wird. Anhand dieses Programms wird gezeigt, wie einfach und konsistent die Verwendung des MDI-API ist. COLORS kann leicht als eine Sammlung von drei verschiedenen, noch zusammenhängenden Programmen gesehen werden. Allein verhalten sie sich wie drei separate Applikationen, obwohl sie die gleiche Fensterfunktion verwenden. Durch das MDI-API werden sie in einem Hauptfenster zusammengelegt.

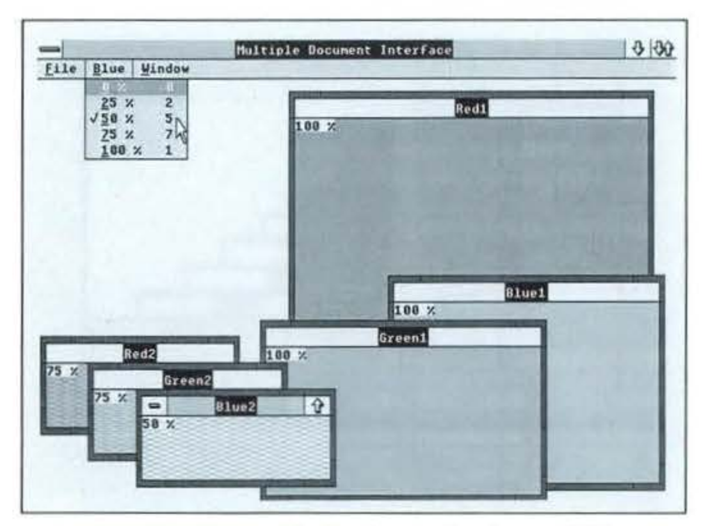


Bild 13c: *Die blauen Tochterfenster besitzen Abkürzungstasten, die im »Blau«-Menü aufgeführt sind (2, 5, 7 oder 1). Mit ihnen kann sehr schnell der Grauwertanteil der blauen Farbe eines Fensters geändert werden. Die Tochterfenster können natürlich in Position und Größe geändert werden.*

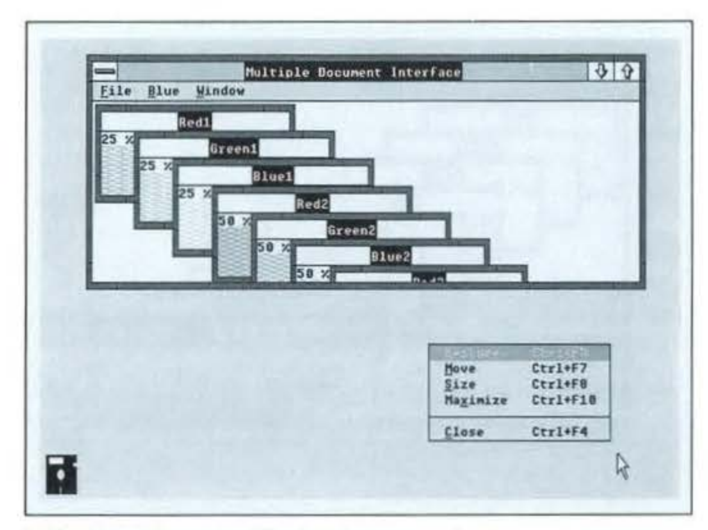


Bild 13d: *Wenn ein Tochterfenster außerhalb des Applikationsfensters von COLORS ist, kann weiterhin über die Tastatur darauf zugegriffen werden. Hierbei wird ein »einsames« Steuermenü angezeigt, genau wie bei Microsoft-Excel (siehe Bild 8).*

Mit der COLORS-Applikation kann man mit Hilfe des »Neu«-Befehls im Menü »Datei« eine Anzahl von roten, grünen und blauen Dokumentfenstern erzeugen. Jedes der Fenster erhält eine sukzessive Nummer und kann über ein gleichnamiges Menü in unterschiedlichen Helligkeiten abgebildet werden. Von den drei Dokumentfenster-Typen hat der Blaue die Besonderheit, daß er Abkürzungstasten besitzt. Durch Drücken der Tasten 0, 1, 2, 5 oder 7 kann die Intensität des blauen Fensterhintergrunds auf 0%, 100%, 25%, 50% oder 75% geändert werden. *Bild 13* zeigt verschiedene Beispiele, wie COLORS aussehen kann.

Wenn mehrere Dokumente im Applikations-Hauptfenster sichtbar sind, kann ein Wechsel von einem Fenster zum anderen über drei verschiedene Mechanismen erfolgen: Selektieren eines Fensters mit der Maus, Verwendung der Tastatur (mit den festgelegten Tasten [Strg][F6] und [Umsch][Strg][F6]) oder Aufruf des Menüs »Fenster« und Auswahl des Namens mit dem gewünschten Dokument. Darüber hinaus kann mit dem »Fenster«-Menü das gerade aktive Dokument verborgen bzw. ein verborgenes wieder angezeigt werden, wie man es von anderen MDI-Applikationen gewohnt ist.

Der Quelltext des Programms besteht aus folgenden fünf Dateien, die in den *Listings 6* bis *10* abgedruckt sind:

COLORS
COLORS.DEF
COLORS.RC
COLORS.H
COLORS.C

Der erste Zugriff auf das MDI-API erfolgt in der MAKE-Datei zum Erzeugen der Applikation (*Listing 6*). Die Übersetzung von `COLORS.C` ist abhängig von den Dateien `MDI.H` und `MDI.LIB`. Weiterhin wird auf die MDI-Bibliothek in der Befehlszeile des Linkers zugegriffen, wodurch COLORS auf alle globalen Routinen des MDI-API zugreifen kann, die wir vorher definiert haben.

Die nächste Datei, die Zugriffe auf den MDI-API enthält, ist die Linker-Definitionsdatei `COLORS.DEF` (*Listing 7*), in der die Funktionen `MdiMsgHook` und `MdiDlgUnhide` als exportiert angegeben worden sind, da sie verschiebbare Eintrittspunkte sind, die vom Windows-System verwendet werden. Wird der Eintrag in `COLORS.DEF` vergessen, treten merkwürdige Fehler auf.

Die dritte Referenz auf das MDI erfolgt in der Ressourcen-Datei `COLORS.RC` (*Listing 8*). Es ist zu beachten, daß die Kopfdatei `MDI.H` eingebunden wird und gegen Ende der Ressourcen-Datei eine Anzahl MDI-bezogener Ressourcen (Dialogfelder etc.) definiert werden. Die erste dieser Ressourcen stellt das MDI-Menü »Fenster« dar, welche als Schablone definiert ist und vom MDI-API an die einzelnen Dokument-Menüleisten angehängt wird sowie automatisch am Menüende um die Namen der angezeigten Dokumente ergänzt wird.

Die nächste Ressource stellt die Tabelle der Abkürzungstasten für die MDI-Tochterfenster dar. Die Tabelle wird automatisch vom API geladen und verwendet, um die Tastaturschnittstelle zum Dokument-Tochterfenster zu implementieren. Die letzte Ressource schließlich definiert das MDI-Dialogfeld für den Befehl »Anzeigen« und wird aufgerufen, nachdem der entsprechende Befehl im »Fenster«-Menü ausgewählt wurde. Als MDI-Entwickler kann man den Stil und Aufbau des Dialogfelds für eigene Zwecke anpassen, man sollte jedoch aufpassen, daß man nicht die Namen und Stilbezeichnungen versehentlich ändert.

Nach der Ressourcen-Datei und der Definitionsdatei `COLORS.H` (*Listing 9*) folgt mit *Listing 10* schließlich `COLORS.C`, welche den gesamten C-Quellcode der COLORS-Applikation enthält und von der Struktur her anderen Windows-Programmen ähnelt. Wie `COLORS.RC` greift auch `COLORS.C` auf die Datei `MDI.H` zu, die außer den benötigten Bezeichnern auch die Prototypen sämtlicher MDI-API-Einsprungfunktionen enthält.

Die erste MDI-bezogene Aktion in `COLORS.C` ist das Anlegen des MDI-Hauptfensters der Applikation durch Aufruf der Funktion `MdiMainCreateWindow` innerhalb von `MainInit`. Dieser Aufruf legt ein leeres Fenster mit dem standardmäßigen Hauptmenü an. Doch bereits kurz nach dem `MdiMainCreateWindow`-Aufruf erfolgt die Ausführung der COLORS-Funktion `ColorCreate`, welche ein neues Dokument-Fenster anlegt, wobei sie die Funktion `MdiChildCreateWindow` verwendet, und weist diesem Fenster die dazugehörende Abkürzungstasten-Tabelle zu. Beim Aufruf von `COLORS` wird dadurch standardmäßig ein rotes Dokumentfenster angelegt.

Nachdem die Oberfläche angelegt und initialisiert worden ist, empfängt die Applikation alle sie betreffenden Nachrichten und verarbeitet sie. Wie bei den meisten Windows-Applikationen erfolgt dies mit einer einfachen Schleife, beginnend mit einem `GetMessage`-Aufruf, dem die Vorübersetzung und anschließende Verteilung für jede eingegangene Nachricht folgt. Hier haben die MDI-Funktionen `MdiGetMessage` und `MdiTranslateAccelerators` jedoch die Windows-API-Funktionen ersetzt.

Der nächste Zugriff auf den MDI-API erfolgt in der Funktion `MainWndProc` von COLORS. Jede Nachricht, die in der Nachrichten-Schleife der Applikation empfangen wurde, wird direkt an die bestimmte Fensterfunktion gesandt. `MainWndProc` empfängt alle Nachrichten, die für das MDI-Hauptfenster bestimmt sind und, da dieses Fenster das einzige mit einem Menü ist, auch alle Nachrichten, die Menüs betreffen.

Die Fensterfunktion des Hauptfensters verarbeitet nur Nachrichten, die die Applikation betreffen. Alle weiteren Nachrichten werden zur Weiterverarbeitung der Funktion `MdiMainDefWindowProc` übergeben. Diese Funktion (im MDI-Code) verarbeitet ihrerseits alle Nachrichten, welche für MDI von Interesse sind und sendet einige dieser Nachrichten weiter an entsprechende Dokument-Tochterfenster. Die anderen Nachrichten werden dem Windows-System mit der Funktion `DefWindowProc` übergeben.

Während des ganzen Prozesses kann das Hauptfenster Nachrichten empfangen, die Befehle angeben, welche in einer der Dokument-Menüleisten aufgerufen wurden. Daher ist es sinnvoll, daß die Dokumentfenster einen gemeinsamen Satz von Menüeinträgen besitzen, die in der Hauptfensterfunktion zentral verarbeitet werden. Bei `COLORS` sind diese Befehle im »Datei«-Menü zusammengefaßt.

Es ist zu beachten, daß lediglich die Menüeinträge von Hauptfenster und Dokumenten konzeptionell getrennt werden müssen, nicht dagegen die Einträge der einzelnen Dokumenten-Menüs. Diese können sich nicht gegenseitig beeinflussen, da nur das aktive Dokument die Befehle seiner aktiven Menüleiste erhält und nicht die von inaktiven Dokumenten.

Die letzten Aufrufe des MDI-API erfolgen in der Fensterfunktion `ColorWndProc`. Diese Funktion wird von allen Dokument-Tochterfenstern gemeinsam benutzt und verarbeitet die dokumentspezifischen Menübefehle und zeichnet den Fensterhintergrund in der entsprechenden Farbe mit der eingestellten Intensität. In `ColorWndProc` wird auf die gesetzte Menüleiste mit `MdiGetMenu` statt mit `GetMenu` zugegriffen. Dies ist wichtig, da das Hauptfenster die Menüleiste enthält und dieses nicht immer das unmittelbare Mutterfenster (*parent window*) des Dokuments sein muß (sonst könnte man auf die Menüleiste mit `GetMenu(GetParent(hWnd))` zugreifen).

Ähnlich wie die Fensterfunktion des Hauptfensters übergibt `ColorWndProc` die meisten Nachrichten der Ersatz-MDI-Nachrichtenverarbeitung, in diesem Fall der Funktion `MdiChildDefWindowProc`. Diese Funktion verarbeitet einige Nachrichten, die MDI betreffen und leitet die anderen über `DefWindowProc` weiter zum System. In bestimmten Situationen werden Nachrichten auch an das Hauptfenster umgeleitet und nicht dem System übergeben.

Wenn Sie `COLORS` erzeugen, experimentieren Sie damit und beobachten Sie dabei, wie die internen Funktionen in verschiedenen Situationen reagieren. Verstecken Sie alle Dokumente oder legen Sie neue an, während sich eines in der Vollbildstellung befindet. Prüfen Sie die Tastaturschnittstelle, indem sie sich von Dokument zu Dokument bewegen, ohne die Maus zu Hilfe zu nehmen.

Nach einer Weile wird Ihnen allmählich bewußt werden, welcher Aufwand im Hintergrund getrieben werden mußte, um die Schnittstelle konsistent zu halten. Doch gerade wegen der sichtbaren Intelligenz wird ein wachsender Overhead vom API aufgebaut. Wenn man schnell zwischen unterschiedlichen Dokument-Fenstern hin und her wechselt, ist dieser Overhead sichtbar. Obwohl dies teilweise auf den recht einfältigen Ansatz bei COLORS in der Nachrichtenverwaltung zurückzuführen ist (alle Nachrichten werden an die Ersatzfensterfunktion weitergegeben) muß dies zu einem Großteil dem MDI-API selbst zugeordnet werden.

Nichtsdestoweniger sollte man sich in Erinnerung rufen, daß die Ziele des hier vorgestellten MDI-API bei der Entwicklung in erster Linie Klarheit und Lesbarkeit waren. Unsere interne Arbeitsversion des API (auf dem die hier vorgestellte Bibliothek beruht) implementiert dagegen die volle MDI-Spezifikation erheblich effizienter als es diese hier tut, sie realisiert die »Fenster«-Menübefehle »Neu«, »Nebeneinander« und »Immer Nebeneinander« und sie kann Dokumentfenster als Sinnbilder darstellen. Die zentralen Strukturen jedoch bleiben die gleichen - mit ein bißchen Leistungssteigerung und Verbesserungen ist die

Basis des vorgestellten API geeignet, Weltklasse-MDI-Applikationen mit unvergleichbarer Leichtigkeit zu unterstützen. Verbunden mit ein wenig Umdenken bei der augenblicklichen Technik bei der Datenverarbeitung werden Sie bei vielen der existierenden Windows-Applikationen die MDI-Benutzerschnittstelle leicht hinzufügen können. Und, vielleicht das Beste daran: Mit dem MDI-API schaffen Sie dies mit wenig Änderungen in Ihrem Quellcode.

MDI und SAA

Zusätzlich zu dem Vorteil der Übertragbarkeit der Benutzeroberfläche durch MDI ist eine der wichtigsten Stützen für Entwickler die Akzeptanz durch die SAA-Spezifikation von IBM. Obwohl ein umfassender Überblick über SAA den Rahmen dieses Artikels sprengen würde wollen wir doch kurz darauf eingehen, wie die SAA-Spezifikation MDI beinflußt.

SAA ist eine Menge ausgewählter Software-Schnittstellen, Konventionen und Protokolle, die als Rahmen für den Entwurf von Applikationen dienen, welche portabel sind und auf drei wichtigen Computer-Standbeinen steht: IBM-System/370, System/3X und Personal Computer.

Ein wichtiger Teil von SAA ist die CUA-Spezifikation. Dieser Standard legt mit einer langen Liste von Regeln und Richtlinien fest, wie ein SAA-gefälliges Benutzerinterface aussehen sollte und wie es verwendet wird. Als Endergebnis entstand ein mehr als 300 Seiten umfassendes Dokument, das bei Ihrem lokalen IBM-Repräsentanten erhältlich ist und in mühsamer Kleinarbeit die SAA-Mensch/Maschinen-Schnittstelle beschreibt.

Warum sind SAA und CUA so bedeutend? Ungeachtet der Tatsache, wie Sie darüber denken, besteht der entscheidende Faktor darin, daß viele große Unternehmen versuchen, auf eine einheitliche Benutzerschnittstelle zu setzen, die eine Vielzahl von Hardware-Plattformen umfaßt.

Dies wird ergänzt durch die Hoffnung, daß Benutzer leicht von Maschine zu Maschine wechseln können, ohne daß sich bei jedem Übergang erneut eine Einarbeitungs-Lernkurve ergibt. Die Unternehmen beginnen, von ihren Verkäufern zunehmend SAA- und CUA-verträgliche Software zu fordern. Microsoft versucht gerade daraus Kapital zu schlagen, indem es ein SAA- und CUA-verträgliches Windows entwickelt (betrifft alle unbenützten Abkürzungstasten).

MDI als integraler Bestandteil der Windows-Strategie von Microsoft paßt in diesen umfassenden Standard. Das Endergebnis - und dies ist der Grund, warum MDI für Sie als Software-Entwickler so bedeutend ist - liegt darin, wenn Sie das MDI-Interface in Ihrer Applikation benutzen (im Gegensatz zu anderen Schemen), daß Ihre Benutzer bereits vertraut sind mit der Schnittstelle und Sie möglicherweise mehr Software verkaufen. Zumindestens sollten Sie einen ausführlicheren Blick auf die SAA- und CUA-Spezifikationen von IBM werfen und sie sich durch den Kopf gehen lassen. Ich persönlich habe lange Zeit mit den Einschränkungen gekämpft, die mir CUA als Entwickler auferlegt, aber ich werde mit ihnen leben können, da sie meine Programme auf eine breitere Basis stellen.

Ich hoffe, daß diese Diskussion Ihnen Ideen und Einblicke gegeben hat, die Ihnen eine Hilfe bei Ihren eigenen Entwicklungen sind. MDI könnte die Antwort auf einige technische Probleme sein, mit denen Sie zur Zeit kämpfen. Wenn Sie sich mit MDI beschäftigen, sollten Sie bedenken, daß es aus der Notwendigkeit heraus entstanden ist, mehrere Fenster innerhalb eines Hauptfensters zu organisieren. Diese Entstehung ist im günstigsten Falle lästig und steht darüber hinaus noch im Widerspruch zur darunterliegenden Hardware. Vielleicht wird in Zukunft etwas ähnliches wie das MDI-API einmal in den Windows- oder OS/2-PM-API eingebaut, was Ihnen und mir große Mühen ersparen dürfte. Bis dahin haben Sie mit diesem Artikel etwas mehr über MDI in der Hand als bisher.

Kevin P. Welch

Kevin P. Welch studierte Computerwissenschaften mit den Schwerpunkten angewandte Mathematik, Robotik und künstliche Intelligenz. Er ist Präsident der Eikon Systems Inc. sowie Doktorand in angewandter Mathematik und schrieb zahlreiche Artikel auf vielen verschiedenen technischen Gebieten.

```
/*
 * MDI.H - Kopfdatei der MDI-Bibliothek
 * (Schnittstelle für MDI-Applikationen)
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Entwickelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 */


/*
 * Programmierer-Interface der Kopfdatei
 */

/* Fensterklassen-Optionen für MDI-Tochterfenster */
#define WS_MDICHILD WS_CAPTION|WS_CHILD|WS_CLIPCHILDREN\
        |WS_VISIBLE|WS_THICKFRAME|WS_CLIPSIBLINGS


/* Menü-Zugriffe für Tochterfenster */
#define MdiGetMenu(w)      GetProp(w,PROP_CHILDMENU)
#define MdiSetAccel(w,a)   SetProp(w,PROP_ACCEL,a)


/* Funktionsaufrufe */
HWND MdiMainCreateWindow
 (LPSTR,LPSTR,DWORD,int,int,int,int,HWND,HMENU,HANDLE,LPSTR);
long MdiMainDefWindowProc(HWND,unsigned,WORD,LONG);
HWND MdiChildCreateWindow
 (LPSTR,LPSTR,DWORD,int,int,int,int,HWND,HMENU,HANDLE,LPSTR);
long MdiChildDefWindowProc(HWND,unsigned,WORD,LONG);
BOOL MdiGetMessage(HWND,LPMSG,HWND,WORD,WORD);
int MdiTranslateAccelerators(HWND,LPMSG);


/* Einträge im Steuermenü eines Tochterfensters */
/* Diese Menüeinträge sollten nicht anderwertig verwendet werden! */
#define IDM_CLOSE        0x3f00
#define IDM_RESTORE      0x3f01
#define IDM_NEXTWINDOW   0x3f02
#define IDM_PREVWINDOW   0x3f03
#define IDM_MOVE         0x3f04
#define IDM_SIZE         0x3f05
#define IDM_MAXIMIZE     0x3f06

/* Einträge im "Fenster"-Menü der Applikation */
#define IDM_NEWWINDOW    0x3f07
#define IDM_ARRANGE      0x3f08
#define IDM_ARRANGEALL   0x3f09
#define IDM_HIDE         0x3f0a
#define IDM_UNHIDE       0x3f0b
#define IDM_FIRSTWIN     0x3f0c
#define WINDOW_POS       8


/*
 * Nichtöffentlicher Teil der Kopfdatei
 */

/* Fensterdaten des MDI-Hauptfensters (Applikation) */
#define PROP_ACTIVE      "MdiParentActiveChild"
#define PROP_COUNT       "MdiParentChildCount"
#define PROP_HIDDEN      "MdiParentHiddenCount"
#define PROP_ZOOM        "MdiParentChildZoomed"
#define PROP_MAINMENU    "MdiParentMenu"
#define PROP_WINDOWMENU  "MdiParentWindowMenu"
#define PROP_CTRLACCEL   "MdiParentChildSysAccel"
#define PROP_SYSMENU     "MdiParentChildSysMenu"
#define PROP_TITLE       "MdiParentTitle"

/* Fensterdatei der MDI-Tochterfenster */
#define PROP_LEFT        "MdiChildRestoreLeft"
#define PROP_TOP         "MdiChildRestoreTop"
#define PROP_WIDTH       "MdiChildRestoreWidth"
#define PROP_HEIGHT      "MdiChildRestoreHeight"
#define PROP_CHILDMENU   "MdiChildMenu"
#define PROP_ACCEL       "MdiChildAccel"
#define PROP_MENUID      "MdiChildId"
#define PROP_ISMDI       "MdiChildWeAreOne"

/* Fensterstil des Listenfelds in Dialogfeld "Anzeigen" */
#define DLGUNHIDE_LB     0x100

/* Rasterbild des Tochterfenster-Steuermenüs */
#define OBM_CLOSE        32767

/* Rückgabewerte von WM_MENUCHAR (stehen nicht in WINDOWS.H) */
#define MC_ABORT         1
#define MC_SELECT        2

/* Statuswerte des Tastatur-Interface für Menüs */
#define POP_NONE         -3
#define POP_MAINSYS      -2
#define POP_CHILDSYS     -1
#define POP_MAIN1ST      0

/*
 * Prototypen der Funktionen
 */
```

Listing 1: *Die Kopfdatei MDI.H*

```
/* MDI2.C */
HWND            MdiCreateChildWindow( int );
void            MdiDestroyChildWindow( HWND );
void            MdiActivateChild( HWND , BOOL );
void            MdiActivateNextChild( HWND );
void            MdiActivatePrevChild( HWND );
void            MdiDeactivateChild( HWND );
void            MdiZoomChild( HWND );
void            MdiRestoreChild( HWND, BOOL );
void            MdiSwitchZoom( HWND, HWND );
HWND            MdiChooseNewActiveChild( HWND );
void            MdiHideChild( HWND );
void            MdiUnhideChild( HWND );
int FAR PASCAL MdiDlgUnhide( HWND, unsigned, WORD, LONG );

/* MDI3.C */
void             MdiZoomMenu( HWND );
void             MdiRestoreMenu( HWND );
void             MdiAppendWindowToMenu( HWND );
void             MdiReinsertWindowInMenu( HWND );
void             MdiRemoveWindowFromMenu( HWND, BOOL );
void             MdiWindowMenu( HWND, HMENU, BOOL );
void             MdiInitSystemMenu( HWND );
HMENU            MdiGetChildSysMenu( void );
HBITMAP          MdiCreateChildSysBitmap( HWND );
void             MdiSetMenuKeyHook( HANDLE );
void             MdiMenuMessageLoopUpdate( HWND );
LONG FAR PASCAL MdiMsgHook( int, WORD, LONG );
void             MdiFreeMenuKeyHook( void );

/* Externe Funktionen (von Windows zur Verfügung gestellt) */
LPSTR FAR PASCAL lstrcpy( LPSTR, LPSTR );
```

Listing 1: *(Ende)*

```
/*
 * MDI1.C - Schnittstelle zur Applikation
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Dieses Modul enthält den gesamten MDI-Code der MDI-Programmier-
 * schnittstelle. Dies umfaßt unter anderem das Erzeugen der Fenster, die
 * Ersatzverarbeitung der Nachrichten und den Code zum Aufruf in der
 * Nachrichtenschleife. Das MDI-Hauptfenster und die MDI-Tochterfenster
 * haben unterschiedliche Funktionen zum Verarbeiten der Nachrichten,
 * während die Nachrichtenschleife fensterunabhängig ist.
 *
 * Entwickelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 */

#include <STDIO.H>
#include <WINDOWS.H>

#include "MDI.H"

/* externe Variablen */
extern int      iCurrentPopup; /* Aktuell angezeigtes Menü */
extern int      iNextPopup;    /* Nächstes Menü */


/*
 * MdiMainCreateWindow( ... ) : HWND;
 *
 *    szClassName    Klassennamen des MDI-Hauptfensters
 *    szTitleName    Namen in der Titelleiste des MDI-Hauptfensters
 *    lStyle         Fensterstil des MDI-Hauptfensters
 *    wLeft          Position des linken Fensterrands
 *    wTop           Position des oberen Fensterrands
 *    wWidth         Breite des Fensters
 *    wHeight        Höhe des Fensters
 *    hwndParent     Mutterfenster des MDI-Hauptfensters
 *    hmenuMain      Bezug auf Menüleiste
 *    hInst          Bezug auf Applikations-Instanz
 *    lpCreateParam  Zeiger auf WM_CREATE-Parameter
 *
 * Lege das MDI-Hauptfenster an. Die Parameter entsprechen exakt denen der
 * Windows-Funktion CreateWindow(). Die MDI-Version der Funktion hält zu-
 * sätzliche Informationen des Hauptfensters in Windows-Besitz-Listen
 * (Property-Lists). Darunter fallen Bezüge auf Menüs, Abkürzungstasten und
 * Rasterbilder sowie Zähler für die Dokument-Fenster.
 */

HWND MdiMainCreateWindow(
   LPSTR       szClassName,
   LPSTR       szTitleName,
   DWORD       lStyle,
   int         wLeft,
   int         wTop,
   int         wWidth,
```

Listing 2: *Die MDI-Einsprünge in MDI1.C*

```
   int         wHeight,
   HWND        hwndParent,
   HMENU       hmenuMain,
   HANDLE      hInst,
   LPSTR       lpCreateParam )

{
   HWND        hwndMain;          /* Bezug auf MDI-Hauptfenster */

   hwndMain = CreateWindow( szClassName,
      szTitleName,
      lStyle,
      wLeft, wTop, wWidth, wHeight,
      hwndParent,
      hmenuMain,
      hInst,
      lpCreateParam );

   /* war es erfolgreich? */
   if ( hwndMain )
   {
      /* setze Systemeingriff für Menü-Nachrichten um Tasten zu verfolgen */
      MdiSetMenuKeyHook( hInst );
      /* initialisiere alle Besitz-Werte des Hauptfensters */
      SetProp( hwndMain, PROP_ACTIVE, NULL );
      SetProp( hwndMain, PROP_COUNT, 0 );
      SetProp( hwndMain, PROP_HIDDEN, 0 );
      SetProp( hwndMain, PROP_ZOOM, FALSE );
      SetProp( hwndMain, PROP_MAINMENU, GetMenu( hwndMain ) );
      SetProp( hwndMain, PROP_WINDOWMENU, LoadMenu(hInst, "MdiMenu") );
      SetProp( hwndMain,PROP_CTRLACCEL,
         LoadAccelerators( hInst, "MdiChildAccel" ) );
      SetProp( hwndMain,PROP_SYSMENU,
         MdiCreateChildSysBitmap( hwndMain ) );
      SetProp( hwndMain, PROP_TITLE, AddAtom( szTitleName ) );
      /* setze Hauptmenüleiste als aktuelle Menüleiste */
      MdiWindowMenu( hwndMain, GetMenu( hwndMain ), TRUE);
   }
   return hwndMain;
}


/*
 * MdiMainDefWindowProc( hwndMain, message, wParam, lParam ) : long;
 *
 *    hwndColors     Bezug des MDI-Hauptfensters
 *    message        übergebene Nachricht
 *    wParam         WORD-Parameter der Nachricht
 *    lParam         LONG-Parameter der Nachricht
 *
 * verarbeitet Nachrichten an das MDI-Hauptfenster, die die Fenster-
 * funktion der MDI-Applikation für das Hauptfenster nicht verarbeitet hat.
 * Zu den wichtigsten Dingen, die die Funktion verarbeitet, gehören:
 * Aktivierung/Deaktivierung des Hauptfensters, Windows-Menünachrichten
 * (insbesondere zum Umschalten zwischen den Tochterfenstern), Verfolgung,
 * welches Menü aufgeklappt ist (wenn überhaupt) und das Vergrößern von
 * Dokumentfenstern, die als Vollbild dargestellt werden sollen.
 */

long MdiMainDefWindowProc(
   HWND        hwndMain,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   FARPROC     lpProc;            /* Prozedur-Instanz für Dialogfelder */
   HANDLE      hInst;             /* Bezug auf aktuelle Instanz */
   HWND        hwndActive;        /* Bezug auf aktuelles Dokument-Fenster */
   HWND        hwndChosen;        /* Bezug auf neu gewähltes Dok.-Fenster */
   HWND        hwndIter;          /* Verwendet beim Ermitteln der Fenster */
   LONG        lStyle;            /* Fensterstil */
   RECT        rcClient;          /* Fensterinhalt des Hauptfensters */

   hwndActive = GetProp( hwndMain, PROP_ACTIVE );
   switch ( message )
   {
   case WM_ACTIVATE:
      /* akt. MDI-Tochterfenster enthält die Aktivierung des Hauptfensters */
      if ( hwndActive )
      {
         switch( wParam )
         {
         case 0:
            /* Deaktiviere */
            MdiDeactivateChild( hwndActive );
            break;
         case 1:
         case 2:
            /* Aktiviere */
            MdiActivateChild( hwndActive, FALSE );
            break;
         }
      }
      break;
   case WM_COMMAND:
      hInst = GetWindowWord( hwndMain, GWW_HINSTANCE );
      switch ( wParam )
      {
      case IDM_UNHIDE:
         /* rufe Dialogfeld "Anzeigen" auf */
         lpProc = MakeProcInstance( MdiDlgUnhide, hInst );
         hwndChosen = DialogBox( hInst, "MdiUnhide", hwndMain, lpProc);
         if ( hwndChosen )
         {
            /* MDI-Tochterfenster wird wieder angezeigt. */
            MdiUnhideChild( hwndChosen );
         }
         FreeProcInstance( lpProc );
         break;
```

Listing 2: *(Fortsetzung)*

```
      default:
         /* prüfe, ob Befehl zum Selektieren eines MDI-Tochterfensters */
         if ( wParam > IDM_FIRSTWIN
            && wParam <= IDM_FIRSTWIN + GetProp( hwndMain, PROP_COUNT))
         {  /* untersuche alle Tochterfenster des MDI-Hauptfensters */
            for ( hwndIter = GetTopWindow( hwndMain );
               hwndIter != NULL;
               hwndIter = GetWindow( hwndIter, GW_HWNDNEXT ) )
            {
               /* überspringe aktives und nicht-MDI Tochterfenster */
               if ( hwndIter == hwndActive
                  || !GetProp( hwndIter, PROP_ISMDI ) )
               {
                  continue;
               }
               /* ist es das Fenster? */
               if ( wParam == GetProp( hwndIter, PROP_MENUID ) )
               {
                  /* zeige ein anderes MDI-Tochterfenster als aktives an */
                  if ( GetProp( hwndMain, PROP_ZOOM ) )
                  {
                     MdiSwitchZoom( hwndIter, hwndActive );
                  }
                  MdiActivateChild( hwndIter, FALSE );
                  break;
               }
            }
         }
         else
         {
            if ( hwndActive )
            {
               /* die anderen WM_COMMAND-Nachrichten enthält das
                  aktive MDI-Tochterfenster */
               return SendMessage(hwndActive, message, wParam, lParam);
            }
         }
         break;
      }
      break;
   case WM_DESTROY:
      /* lösche alle angelegten Objekte */
      DeleteObject( GetProp( hwndMain, PROP_SYSMENU ) );
      DeleteAtom( GetProp( hwndMain, PROP_TITLE ) );
      MdiFreeMenuKeyHook( );
      break;
   case WM_INITMENU:
      /* irgendetwas zu verbergen? Mache "Verbergen"-Eintrag entspr. grau */
      EnableMenuItem( ( HMENU ) wParam,
         IDM_HIDE,
         hwndActive ? MF_ENABLED : MF_GRAYED );
      /* irgendwelche verstecke Fenster? Mache "Anzeigen" entspr. grau */
      EnableMenuItem( ( HMENU ) wParam,
         IDM_UNHIDE,
         GetProp( hwndMain, PROP_HIDDEN ) ? MF_ENABLED : MF_GRAYED );
      /* möglicherweise benötigt das MDI-Tochterfenster die Nachricht auch */
      if ( hwndActive )
      {
         SendMessage( hwndActive, message, wParam, lParam );
      }
      break;
   case WM_INITMENUPOPUP:
      /* verfolge, welches Menü aufgeklappt ist, sodaß beim Drücken der
         linken oder rechten Richtungstaste klar ist, welches das nächste Menü ist */
      if ( HIWORD( lParam ) )
      {
         iCurrentPopup = POP_MAINSYS;
      }
      else
      {
         iCurrentPopup = LOWORD( lParam );
         if ( GetProp( hwndMain, PROP_ZOOM ) )
            iCurrentPopup--;
      }
      if ( hwndActive )
      {
         /* möglicherweise benötigt das MDI-Tochterfenster die Nachricht */
         SendMessage( hwndActive, message, wParam, lParam );
      }
      break;
   case WM_MENUCHAR:
      if ( wParam == '-' )
      {
         /* stelle Steuermenü des aktiven MDI-Tochterfensters dar */
         if ( ( LOWORD( lParam ) & MF_POPUP ) == 0 && hwndActive )
         {
            /* ist das MDI-Tochterfenster im Vollbild? */
            if ( GetProp( hwndMain, PROP_ZOOM ) )
            {
               /* MDI-Tochterfester-Steuermenü ist in Haupt-Menüleiste */
               return MAKELONG( 0, MC_SELECT );
            }
            iNextPopup = POP_CHILDSYS;
            /* als Ergebnis wird zunächst kein Menü angezeigt, aber da
               <iNextPopup> gesetzt wird, wird der nächste Nachrichten-Aufruf
               das Steuermenü des Tochterfensters aufrufen */
            return MAKELONG( 0, MC_ABORT );
         }
         break;
      }
      break;
   case WM_SETFOCUS:
      /* setze den Fokus des MDI-Hauptfensters als Fokus des aktiven
         MDI-Tochterfensters */
      if ( hwndActive )
      {
         SetFocus( hwndActive );
      }
      break;
```

Listing 2: *(Fortsetzung)*

```
   case WM_SIZE:
      if ( wParam != SIZEICONIC )
      {
         /* verändere Größe des MDI-Tochterfensters wenn Vollbild */
         if ( GetProp( hwndMain, PROP_ZOOM ) )
         {
            GetClientRect( hwndMain, &rcClient );
            lStyle = GetWindowLong( hwndActive, GWL_STYLE );
            AdjustWindowRect( &rcClient, lStyle, FALSE );
            MoveWindow( hwndActive,
               rcClient.left,
               rcClient.top,
               rcClient.right - rcClient.left,
               rcClient.bottom - rcClient.top,
               TRUE );
         }
      }
      break;
   }
   return DefWindowProc( hwndMain, message, wParam, lParam );
}

/*
 * MdiChildCreateWindow( ... ) : HWND;
 *
 *    szClassName    Klassennamen des MDI-Tochterfensters
 *    szTitleName    Namen in der Titelleiste des MDI-Tochterfensters
 *    lStyle         Fensterstil des MDI-Tochterfensters
 *    wLeft          Position des linken Fensterrands
 *    wTop           Position des oberen Fensterrands
 *    wWidth         Breite des Fensters
 *    wHeight        Höhe des Fensters
 *    hwndMain       Bezug auf MDI-Hauptfensters
 *    hmenuChild     Bezug auf Menüleiste des Dokuments
 *    hInst          Bezug auf Applikations-Instanz
 *    lpCreateParam  Zeiger auf WM_CREATE-Parameter
 *
 * Lege ein MDI-Dokument-Tochterfenster an. Die Parameter entsprechen exakt
 * denen der Windows-Funktion CreateWindow(), außer daß <hmenuChild> der
 * Bezug der gültigen Menüleiste ist, wogegen CreateWindow() stattdessen die
 * Tochter-Identifikation erwartet. Die MDI-Version der Funktion hält zu-
 * sätzliche Informationen des Hauptfensters in Windows-Besitz-Listen
 * (Property-Lists). Darunter fallen Bezüge auf Menüs und Abkürzungstasten
 * und beispielsweise ein Flag, ob das Tochterfenster tatsächlich ein
 * MDI-Dokument ist.
 */

HWND MdiChildCreateWindow(
   LPSTR       szClassName,
   LPSTR       szTitleName,
   DWORD       lStyle,
   int         wLeft,
   int         wTop,
   int         wWidth,
   int         wHeight,
   HWND        hwndMain,
   HMENU       hmenuChild,
 - HANDLE      hInst,
   LPSTR       lpCreateParam )
{
   char        szTitle[128];     /* Titelleisten-Text des Dokuments */
   char        szCompose[128];   /* Verwendet beim Zusammenbau von Texten */
   int         iCount;           /* Tochterfenster-Identifikation */
   int         wLeftAct = wLeft;       /* Position des linken Fensterrands */
   int         wTopAct = wTop;         /* Position des oberen Fensterrands */
   int         wWidthAct = wWidth;     /* Breite des Fensters */
   int         wHeightAct = wHeight;   /* Höhe des Fensters */
   HWND        hwndActive;       /* Bezug auf aktives Dokument-Fenster */
   HWND        hwndChild;        /* Bezug auf Dokumentfenster */
   RECT        rcClient;         /* Fensterinhalt des MDI-Hauptfensters */

   /* Berechne Größe und Position */
   iCount = GetProp( hwndMain, PROP_COUNT );
   if ( wLeftAct == CW_USEDEFAULT )
      wLeftAct = 10 + 15 * iCount;
   if ( wTopAct == CW_USEDEFAULT )
      wTopAct = 10 + 15 * iCount;
   if ( wWidthAct == CW_USEDEFAULT )
      wWidthAct = 200;
   if ( wHeightAct == CW_USEDEFAULT )
      wHeightAct = 75;
   /* überprüfe, ob es innerhalb des Fensterinhalts steht */
   GetClientRect( hwndMain, &rcClient );
   if ( wLeft == CW_USEDEFAULT )
      while ( wLeftAct > rcClient.right )
         wLeftAct -= rcClient.right;
   if ( wTop == CW_USEDEFAULT )
      while ( wTopAct > rcClient.bottom )
         wTopAct -= rcClient.bottom;
   /* erzeuge Fenster */
   hwndChild = CreateWindow( szClassName,
      szTitleName,
      lStyle,
      wLeftAct, wTopAct, wWidthAct, wHeightAct,
      hwndMain,
      ( HMENU ) iCount,
      hInst,
      lpCreateParam );
   /* war es erfolgreich? */
   if ( hwndChild )
   {
      /* neues Tocherfenster */
      SetProp( hwndMain, PROP_COUNT, ++iCount );
      /* initialisiere wichtige Daten */
      SetProp( hwndChild, PROP_CHILDMENU, hmenuChild );
      SetProp( hwndChild, PROP_ACCEL, 0 );
      SetProp( hwndChild, PROP_MENUID, IDM_FIRSTWIN + iCount );
      SetProp( hwndChild, PROP_ISMDI, TRUE );
```

Listing 2: *(Fortsetzung)*

```
      /* berücksichtige Änderungen im Hauptfenster-Menü */
      MdiAppendWindowToMenu( hwndChild );
      /* setze erzeugtes Tochterfenster als aktives Fenster */
      hwndActive = GetProp( hwndMain, PROP_ACTIVE );
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         MdiSwitchZoom( hwndChild, hwndActive );
      }
      MdiActivateChild( hwndChild, FALSE );
      /* stelle geänderte Menüleiste neu dar */
      DrawMenuBar( hwndMain );
   }
   return hwndChild;
}

/*
 * MdiChildDefWindowProc( hwndChild, message, wParam, lParam ) : long;
 *
 *    hwndColors     Bezug des MDI-Dokuments
 *    message        übergebene Nachricht
 *    wParam         WORD-Parameter der Nachricht
 *    lParam         LONG-Parameter der Nachricht
 *
 * verarbeitet Nachrichten an ein MDI-Tochterfenster, die die Fenster-
 * funktion der MDI-Applikation für das Hauptfenster nicht verarbeitet
 * hat.
 * Zu den wichtigsten Dingen, die die Funktion verarbeitet, gehören:
 * Aufruf des Tochterfenster-Steuermenüs, Fenster-Aktivierung, Fenster-
 * Vernichtung und die Umwandlung von <Alt>+'-' in den Aufruf des Steuer-
 * menüs.
 */

long MdiChildDefWindowProc(
   HWND        hwndChild,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */
   PAINTSTRUCT Paint;            /* Zeichnen-Struktur */

   hwndMain = GetParent( hwndChild );
   switch ( message )
   {
   case WM_CLOSE:
      /* Fenster wird geschlossen */
      MdiDestroyChildWindow( hwndChild );
      break;
   case WM_COMMAND:
      switch( wParam )
      {
      case IDM_HIDE:
         /* Fenster wird verborgen */
         MdiHideChild( hwndChild );
         break;
      case IDM_RESTORE:
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_RESTORE, lParam );
         break;
      case IDM_MOVE:
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_MOVE, lParam );
         break;
      case IDM_SIZE:
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_SIZE, lParam );
         break;
      case IDM_MAXIMIZE:
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_MAXIMIZE, lParam );
         break;
      case IDM_PREVWINDOW:
         /* Dieser Aufruf erfolgt nur über die Tastatur */
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_PREVWINDOW, lParam);
         break;
      case IDM_NEXTWINDOW:
         /* Dieser Aufruf erfolgt nur über die Tastatur */
         SendMessage( hwndChild, WM_SYSCOMMAND, SC_NEXTWINDOW, lParam);
         break;
      }
      break;
   case WM_INITMENU:
      MdiInitSystemMenu( hwndChild );
      break;
   case WM_MENUCHAR:
      /* Würde <Alt>+'-' Taste gedrückt als kein Menü aufgeklappt war? */
      if ( ( LOWORD( lParam ) & MF_POPUP ) == 0 )
      {
         if ( wParam == '-' )
         {
            /* Alt-Minus bleibt aktiv im Steuermenü */
            return MAKELONG( 0, MC_SELECT );
         }
         else
         {
            /* Andere <Alt>-Taste gedrückt: Verwaltet das MDI-Hauptfenster */
            SendMessage( hwndMain,
               WM_SYSCOMMAND,
               SC_KEYMENU,
               ( DWORD ) wParam );

            return MAKELONG( 0, MC_ABORT );
         }
      }
      break;
   case WM_MENUSELECT:
      iCurrentPopup = POP_CHILDSYS;
      break;
   case WM_MOUSEACTIVATE:
      MdiActivateChild( hwndChild, FALSE );
      break;
   case WM_SYSCOMMAND:
      switch( wParam & 0xFFF0 )
```

Listing 2: *(Fortsetzung)*

```
      {
      case SC_KEYMENU:
         if ( GetProp( hwndMain, PROP_ZOOM ) || LOWORD(lParam) != '-' )
         {
            /* Schicke andere Alt-Tasten-Nachrichten an Hauptfenster */
            return SendMessage(hwndMain,WM_SYSCOMMAND, wParam, lParam);
         }
         break;
      case SC_MAXIMIZE:
         if ( GetProp( hwndMain, PROP_ZOOM ) )
         {
            SetProp( hwndMain, PROP_ZOOM, FALSE );
            MdiRestoreChild( hwndChild, TRUE );
         }
         else
         {
            SetProp( hwndMain, PROP_ZOOM, TRUE );
            MdiZoomChild( hwndChild );
         }
         return 0L;
      case SC_RESTORE:
         if ( GetProp( hwndMain, PROP_ZOOM ) )
         {
            SetProp( hwndMain, PROP_ZOOM, FALSE );
            MdiRestoreChild( hwndChild, TRUE );
         }
         return 0L;
      case SC_NEXTWINDOW:
         /* Dieser Aufruf erfolgt nur über die Tastatur */
         MdiActivateNextChild( hwndChild );
         return 0L;
      case SC_PREVWINDOW:
         /* Dieser Aufruf erfolgt nur über die Tastatur */
         MdiActivatePrevChild( hwndChild );
         return 0L;
      }
      break;
   }
   return DefWindowProc( hwndChild, message, wParam, lParam );
}


/*
 * MdiGetMessage( hwndMain, lpMsg, hWnd, wMin, wMax ) : BOOL;
 *
 *    hwndMain       Bezug des MDI-Hauptfensters
 *    lpMsg          Nachrichten-Struktur zum Empfang der Nachricht
 *    hWnd           Sind Nachrichten nur für bestimmtes Fenster?
 *    wMin           Gibt es einen Mindest-Nachrichtenwert?
 *    wMax           Gibt es einen Maximal-Nachrichtenwert?
 *
 * Liest eine normale Fensternachricht erst dann ein, nachdem geprüft wurde,
 * ob über die Tastatur auf die Menüs zugegriffen wird.
 */

BOOL MdiGetMessage(
   HWND        hwndMain,
   LPMSG       lpMsg,
   HWND        hWnd,
   WORD        wMin,
   WORD        wMax )
{
   /* Realisiere Tastatur-Interface von Menü */
   MdiMenuMessageLoopUpdate( hwndMain );
   /* Lese anschließend nächste Nachricht */
   return GetMessage( lpMsg, hWnd, wMin, wMax );
}


/*
 * MdiTranslateAccelerators( hwndMain, lpMsg ) : int
 *
 *    hwndMain       Bezug des MDI-Hauptfensters
 *    lpMsg          Nachrichten-Struktur der aktuellen Nachricht
 *
 * Übersetze die Nachricht über eine der beiden folgenden
 * Abkürzungstasten-Tabellen:
 *  1) Dokument-Steuermenü (z.B. <ctrl>+<F4>)
 *  2) Die eigene Abkürzungstasten-Tabelle des Dokuments, die mit der
 *     Funktion MdiSetAccelerator() gesetzt wurde.
 */

int MdiTranslateAccelerators(
   HWND        hwndMain,
   LPMSG       lpMsg )
{
   int         iResult = 0;      /* Ergebnis der letzten Übersetzung */
   HANDLE      hAccel;           /* Eine der beiden Abkürzungs-Tabellen */
   HWND        hwndChild;        /* Bezug auf Dokument-Tochterfenster */

   /* Bestimme aktives Dokument, falls vorhanden */
   hwndChild = GetProp( hwndMain, PROP_ACTIVE );
   if ( hwndChild )
   {
      /* Prüfe, ob Steuermenü-Abkürzungstaste */
      hAccel = GetProp( hwndMain, PROP_CTRLACCEL );
      iResult = TranslateAccelerator( hwndMain, hAccel, lpMsg );
      if ( !iResult )
      {
         /* Prüfe, ob dokumenteigene Abkürzungstaste */
         hAccel = GetProp( hwndChild, PROP_ACCEL );
         if ( hAccel )
         {
            iResult = TranslateAccelerator( hwndMain, hAccel, lpMsg );
         }
      }
   }
   return iResult;
}
```

Listing 2: *(Ende)*

```
/*
 * MDI2.C - Child Window Routines
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Dieses Modul enthält den gesamten MDI-Code zur Verwaltung der Aktivie-
 * rungen zwischen den MDI-Dokument-Fenstern. Es enthält auch die Vollbild-
 * darstellung und Normalisierung der Dokumente und insbesondere das Um-
 * schalten zwischen den Fenstern. Weiterhin befindet sich hier der Code
 * für das Dialogfeld des Befehls "Anzeigen".
 *
 * Entwickelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 */

#include <STDIO.H>
#include <STRING.H>
#include <WINDOWS.H>

#include "MDI.H"


/*
 * MdiDestroyChildWindow( hwndChild ) : void;
 *
 *    hwndChild      Handle of document window
 *
 * Diese Funktion wird aufgerufen, wenn ein Dokument-Tochterfenster
 * geschlossen wird. Sie nimmt ein anderes Dokumentfenster und aktiviert es.
 * Wenn kein anderes Dokumentfenster verfügbar (sichtbar) ist, wird die
 * Original-Menüleiste des MDI-Hauptfensters angezeigt und der Fokus dem
 * Hauptfenster zugewiesen.
 */

void MdiDestroyChildWindow(
   HWND        hwndChild )
{
   HMENU       hmenuMain;        /* Bezug auf MDI-Hauptfenster-Menüleiste */
   HMENU       hmenuOld;         /* Bezug auf Dokument-Menü */
   HWND        hwndNext;         /* Nächstes aktives Dokumentfenster*/
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */

   /* Entferne das Dokument aus dem "Fenster"-Menü */
   MdiRemoveWindowFromMenu( hwndChild, TRUE );
   /* Wähle ein neues MDI-Tochterfenster aus, das aktiv werden soll */
   hwndMain = GetParent( hwndChild );
   hwndNext = MdiChooseNewActiveChild( hwndChild );
   if ( hwndNext )
   {
      /* Aktiviere dieses MDI-Tochterfenster */
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         MdiSwitchZoom( hwndNext, hwndChild );
      }
      MdiActivateChild( hwndNext, FALSE );
   }
   else
   {
      /* Kein anderes MDI-Tochterfenster ist aktiv */
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         SetProp( hwndMain, PROP_ZOOM, FALSE );
         MdiRestoreChild( hwndChild, FALSE );
      }
      /* Zurück zur Menüleiste ohne "Window"-Menü */
      hmenuOld = GetMenu( hwndMain );
      hmenuMain = GetProp( hwndMain, PROP_MAINMENU );
      MdiWindowMenu( hwndMain, hmenuMain, TRUE );
      SetMenu( hwndMain, hmenuMain );
      MdiWindowMenu( hwndMain, hmenuOld, FALSE );
      SetProp( hwndMain, PROP_ACTIVE, NULL );
      /* Fokus wieder zum MDI-Hauptfenster */
      SetFocus( hwndMain );
   }
   /* Zeige Veränderungen an */
   DrawMenuBar( hwndMain );
}


/*
 * MdiActivateChild( hwndChild, bHidden ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *    bHidden        War Dokument-Fenster vorher verborgen?
 *
 * Aktiviere das angegebene Dokument-Fenster. Es erhält die gültige
 * Titelleiste, es wird als oberstes Tochterfenster angezeigt, sein
 * Menü wird als MDI-Hauptfenster-Menü gesetzt etc.
 */

void MdiActivateChild(
   HWND        hwndChild,
   BOOL        bHidden )
{
   HMENU       hmenuOld;         /* Vorherige Dokument-Menüleiste */
   HMENU       hmenuNew;         /* Menüleiste dieses Dokuments */
   HMENU       hmenuWindow;      /* "Fenster"-Menü */
   HWND        hwndActive;       /* Bezug auf vorheriges Dokument */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */
   LONG        lStyle;           /* Fensterstil des Dokument-Fensters */
```

Listing 3: *Aktivierung und Umschalten der Dokumente*

```
   /* Prüfe, ob dieses Tochterfenster bereits aktiv */
   hwndMain = GetParent( hwndChild );
   lStyle = GetWindowLong( hwndChild, GWL_STYLE );
   if ( ( lStyle & WS_MAXIMIZEBOX ) == 0 )
   {  /* Ändere die Farbe der Titelleiste */
      SendMessage( hwndChild, WM_NCACTIVATE, TRUE, 0L );
      /* Füge Steuermenü und Vollbildfeld der Titelleiste hinzu */
      SetWindowLong( hwndChild,
         GWL_STYLE,
         lStyle | WS_MAXIMIZEBOX | WS_SYSMENU );
      /* Mache die Änderungen sichtbar */
      if ( bHidden )
      {
         BringWindowToTop( hwndChild );
         ShowWindow( hwndChild, SW_SHOW );
      }
      else
      {  /* Setze Fenster als oberstes und zeichne Rahmen neu */
         SetWindowPos( hwndChild,
            NULL,0, 0, 0, 0,
            SWP_DRAWFRAME | SWP_NOMOVE | SWP_NOSIZE );
      }
      /* Aktiviere das Tochterfenster */
      hwndActive = GetProp( hwndMain, PROP_ACTIVE );
      if ( hwndActive != hwndChild )
      {
         /* Entferne Aktivierungsdetails vom vorher aktiven Fenster */
         if ( hwndActive && hwndActive != hwndChild )
         {
            MdiDeactivateChild( hwndActive );
         }
         /* Setze gültige Menüleiste für dieses MDI-Tochterfenster */
         hmenuNew = GetProp( hwndChild, PROP_CHILDMENU );
         hmenuOld = GetMenu( hwndMain );
         MdiWindowMenu( hwndMain, hmenuNew, TRUE );
         SetMenu( hwndMain, hmenuNew );
         MdiWindowMenu( hwndMain, hmenuOld, FALSE );
         DrawMenuBar( hwndMain );
         /* Das Tochterfenster ist jetzt aktiv */
         SetProp( hwndMain, PROP_ACTIVE, hwndChild );
      }
      /* Setze das neue Fenster in der Liste des "Fenster"-Menüs */
      hmenuWindow = GetProp( hwndMain, PROP_WINDOWMENU );
      CheckMenuItem( hmenuWindow,
         GetProp( hwndChild, PROP_MENUID ),
         MF_CHECKED | MF_BYCOMMAND );
      /* Das neue Tochterfenster sollte den Eingabefokus haben */
      if ( GetFocus( ) != hwndChild )
      {
         SetFocus( hwndChild );
      }
   }
}


/*
 * MdiActivateNextChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf aktuelles Dokument-Fenster
 *
 * Aktiviere das nächste Fenster in der internen Fenster-Liste.
 * Diese Funktion wird aufgerufen, wenn <Strg>+<F6> eingegeben wird.
 */

void MdiActivateNextChild(
   HWND        hwndChild )
{
   HWND        hwndActive;       /* Bezug auf vorheriges Dokument */
   HWND        hwndIter;         /* Bezug bei der Fenster-Auswahl */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */

   for ( hwndIter = GetWindow( hwndChild, GW_HWNDNEXT );
      hwndIter != NULL;
      hwndIter = GetWindow( hwndIter, GW_HWNDNEXT ) )
   {  /* Nimm nächstes sichtbares MDI-Tochterfenster in der
         internen Fensterliste des Windows-Systems.
      */
      if ( GetProp(hwndIter, PROP_ISMDI) && IsWindowVisible(hwndIter) )
      {
         break;
      }
   }
   /* Wurde ein anderes MDI-Tochterfenster gefunden? */
   if ( hwndIter )
   {
      hwndMain = GetParent( hwndIter );
      hwndActive = GetProp( hwndMain, PROP_ACTIVE );
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         /* Aktiviere das neue Fenster */
         MdiSwitchZoom( hwndIter, hwndActive );
         MdiActivateChild( hwndIter, FALSE );
         /* Setze Fenster ans Ende der internen Fensterliste */
         SetWindowPos( hwndChild,
            ( HWND ) 1,
            0, 0, 0, 0,
            SWP_NOMOVE | SWP_NOSIZE | SWP_NOACTIVATE );
      }
      else
      {  /* Setze Fenster ans Ende der internen Fensterliste */
         SetWindowPos( hwndChild,
            ( HWND ) 1,
            0, 0, 0, 0,
            SWP_NOMOVE | SWP_NOSIZE | SWP_NOACTIVATE );
         /* Aktiviere das neue Fenster */
         MdiActivateChild( hwndIter, FALSE );
      }
   }
}
```

Listing 3: *(Fortsetzung)*

```
/*
 * MdiActivatePrevChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf aktuelles Dokument-Fenster
 *
 * Aktiviere das vorherige Fenster in der internen Fenster-Liste.
 * Diese Funktion wird aufgerufen, wenn <Strg>+<Umsch>+<F6> eingegeben wird.
 */

void MdiActivatePrevChild(
   HWND        hwndChild )
{
   HWND        hwndActive;       /* Bezug auf vorheriges Dokument */
   HWND        hwndIter;         /* Bezug bei der Fenster-Auswahl */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */

   for ( hwndIter = GetWindow( hwndChild, GW_HWNDLAST );
      hwndIter != NULL;
      hwndIter = GetWindow( hwndIter, GW_HWNDPREV ) )
   {
      /* Nimm nächstes sichtbares MDI-Tochterfenster in der
         internen Fensterliste des Windows-Systems. */
      if ( GetProp(hwndIter, PROP_ISMDI) && IsWindowVisible(hwndIter) )
      {
         break;
      }
   }
   /* Wurde ein anderes MDI-Tochterfenster gefunden? */
   if ( hwndIter )
   {
      /* Aktiviere das neue Fenster */
      hwndMain = GetParent( hwndChild );
      hwndActive = GetProp( hwndMain, PROP_ACTIVE );
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         MdiSwitchZoom( hwndIter, hwndActive );
      }
      MdiActivateChild( hwndIter, FALSE );
   }
}

/*
 * MdiDeactivateChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *
 * Stelle das angegebene Dokumentfenster zurück, weil entweder irgendein
 * anderes Dokumentfenster aktiviert wird oder die Applikation den Eingabe-
 * fokus verliert.
 */

void MdiDeactivateChild(
   HWND        hwndChild )
{
   LONG        lStyle;           /* Fensterstil des Dokument-Fensters */

   /* prüfe, ob das MDI-Tochterfenster bereits deaktiviert ist */
   lStyle = GetWindowLong( hwndChild, GWL_STYLE );
   if ( lStyle & WS_MAXIMIZEBOX )
   {
      /* Wechsele die Farbe der Titelleiste */
      SendMessage( hwndChild, WM_NCACTIVATE, FALSE, 0L );
      /* Entferne Steuermenü und Vollbildfeld aus der Titelleiste */
      SetWindowLong( hwndChild,
         GWL_STYLE,
         lStyle & ~( WS_MAXIMIZEBOX | WS_SYSMENU ) );
      SetWindowPos( hwndChild,
         NULL,
         0, 0, 0, 0,
         SWP_DRAWFRAME  | SWP_NOACTIVATE  | SWP_NOMOVE
                        | SWP_NOSIZE      | SWP_NOZORDER );
   }
   /* Entferne Marke aus der Liste im "Fenster"-Menü */
   CheckMenuItem( GetProp( GetParent( hwndChild ), PROP_WINDOWMENU ),
      GetProp( hwndChild, PROP_MENUID ),
      MF_UNCHECKED | MF_BYCOMMAND );
}


/*
 * MdiZoomChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *
 * Bringe das angegebene Dokument in den gesamten Zeichenbereich des
 * MDI-Hauptfensters. Standardmäßig wird der Rahmen und die Titelleiste
 * in den Zeichenbereich gebracht. Diese Funktion dagegen setzt den
 * Zeichenbereich des Dokuments in den Zeichenbereich des MDI-Hauptfen-
 * sters und den Titel des Dokuments in die Titelleiste des Hauptfensters.
 */

void MdiZoomChild(
   HWND        hwndChild )
{
   char        szTitle[128];     /* Text für Titelleiste */
   char        szMain[128];      /* Text für Nicht-Vollbild-Titelleiste */
   char        szChild[128];     /* Text für Dokument-Titelleiste */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */
   LONG        lStyle;           /* Fensterstil des Dokumentfensters */
   RECT        rcClient;         /* Fensterinhalt des MDI-Hauptfensters */

   /* Wechsle die Menüleiste */
   hwndMain = GetParent( hwndChild );
   MdiZoomMenu( hwndChild );
   DrawMenuBar( hwndMain );
   /* Ändere die Titelleiste */
   GetAtomName( GetProp( hwndMain, PROP_TITLE ),
      szMain,
      sizeof(szMain) );
```

Listing 3: *(Fortsetzung)*

```
   GetWindowText( hwndChild, szChild, sizeof( szChild ) );
   sprintf( szTitle, "%s - %s", szMain, szChild );
   SetWindowText( hwndMain, szTitle );
   /* Sichere die gegenwärtige Position innerhalb des Zeichenbereichs */
   GetWindowRect( hwndChild, &rcClient );
   ScreenToClient( hwndMain, ( LPPOINT ) &rcClient+0 );
   ScreenToClient( hwndMain, ( LPPOINT ) &rcClient+1 );
   SetProp( hwndChild, PROP_LEFT, rcClient.left );
   SetProp( hwndChild, PROP_TOP, rcClient.top );
   SetProp( hwndChild, PROP_WIDTH, rcClient.right - rcClient.left );
   SetProp( hwndChild, PROP_HEIGHT, rcClient.bottom - rcClient.top );
   /* Positioniere das Fenster so, daß sein Zeichenbereich den gesamten
      Zeichenbereichs des MDI-Hauptfensters belegt */
   GetClientRect( hwndMain, &rcClient );
   lStyle = GetWindowLong( hwndChild, GWL_STYLE );
   AdjustWindowRect( &rcClient, lStyle, FALSE );
   SetWindowPos( hwndChild,
      ( HWND ) NULL,
      rcClient.left, rcClient.top,
      rcClient.right - rcClient.left, rcClient.bottom - rcClient.top,
      0 );
   /* Setze den Fensterstil "Vollbild" */
   SetWindowLong( hwndChild, GWL_STYLE, lStyle | WS_MAXIMIZE );
}


/*
 * MdiRestoreChild( hwndChild, bHidden ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *    bShowMove      Soll das Fenster angezeigt bleiben oder verborgen werden
 *
 * Bringe das angegebene Dokument von der Vollbilddarstellung zurück in
 * die Darstellung zuvor. Das Dokument wird mit der Größe vor der Voll-
 * bilddarstellung an die entsprechende Position plaziert. Diese Werte werden
 * aus der Besitzliste ermittelt. Weiterhin wird die Titelleiste des MDI-
 * Hauptfensters wiederhergestellt.
 */

void MdiRestoreChild(
   HWND         hwndChild,
   BOOL         bShowMove )
{
   char         szMain[128];       /* Text der Nicht-Vollbild-Titelleiste */
   HWND         hwndMain;          /* Bezug auf das MDI-Hauptfenster */
   LONG         lStyle;            /* Fensterstil des Dokument-Fensters */

   /* Wechsele die Menüleiste */
   hwndMain = GetParent( hwndChild );
   MdiRestoreMenu( hwndChild );
   DrawMenuBar( hwndMain );
   /* Stellt Titelleiste des Applikationsfensters wieder her */
   GetAtomName( GetProp( hwndMain, PROP_TITLE ),
      szMain,
      sizeof( szMain ) );
   SetWindowText( hwndMain, szMain );
   /* Setze Fensterstil auf normale Größe */
   lStyle = GetWindowLong( hwndChild, GWL_STYLE );
   SetWindowLong( hwndChild, GWL_STYLE, lStyle & ~WS_MAXIMIZE );
   /* Verberge Fenster falls gewünscht */
   if ( !bShowMove )
   {
      ShowWindow( hwndChild, SW_HIDE );
   }
   /* Bewege Fenster zu der ursprünglichen Position in entsprechender Größe */
   /* Setze es auch an den Beginn der internen Fensterliste des Systems */
   SetWindowPos( hwndChild,
      ( HWND ) NULL,
      GetProp( hwndChild, PROP_LEFT ),
      GetProp( hwndChild, PROP_TOP ),
      GetProp( hwndChild, PROP_WIDTH ),
      GetProp( hwndChild, PROP_HEIGHT ),
      0 );
}


/*
 * MdiSwitchZoom( hwndNew, hwndCur ) : void;
 *
 *    hwndChild      Bezug auf neues Dokument-Fenster
 *    hwndCur        Bezug auf bisheriges Dokument-Fenster
 *
 * Diese Routine wechselt die Aktivierung zwischen zwei Dokument-Fenstern,
 * wenn das bisherige Dokument-Fenster als Vorbild darstellt wird. Bei jedem
 * Wechsel ist die Reihenfolge der Aktivierung wichtig. Wenn die Dokumente
 * Vollbilder sind, ist es wichtig, nicht dem gewöhnlichen Ablauf zu folgen,
 * sonst ist es möglich, daß man kurzzeitig beim Wechseln ein Nicht-Vollbild-
 * Fenster sieht, bevor das neue Dokument den Zeichenbereich ganz ausfüllt.
 */

void MdiSwitchZoom(
   HWND         hwndNew,
   HWND         hwndCur )
{
   char         szTitle[128];      /* Text für die Titelleiste */
   char         szMain[128];       /* Text für die Nicht-Vollbild-Titelleiste */
   char         szChild[128];      /* Text für Dokument-Titelleiste */
   HWND         hwndMain;          /* Bezug auf MDI-Hauptfenster */
   LONG         lStyle;            /* Fensterstil des Dokumentfensters */
   RECT         rcClient;          /* Fensterinhalt des MDI-Hauptfensters */

   /* Wechsele die Titelleiste */
   hwndMain = GetParent( hwndCur );
   GetAtomName( GetProp( hwndMain, PROP_TITLE ),
      szMain,
      sizeof(szMain) );
   GetWindowText( hwndNew, szChild, sizeof( szChild ) );
   sprintf( szTitle, "%s - %s", szMain, szChild );
   SetWindowText( hwndMain, szTitle );
```

Listing 3: *(Fortsetzung)*

```
   /* NEUES Dokument-Fenster */
   /* Ermittle die Position dieses Fensters im Zeichenbereich */
   GetWindowRect( hwndNew, &rcClient );
   ScreenToClient( hwndMain, ( LPPOINT ) &rcClient+0 );
   ScreenToClient( hwndMain, ( LPPOINT ) &rcClient+1 );
   SetProp( hwndNew, PROP_LEFT, rcClient.left );
   SetProp( hwndNew, PROP_TOP, rcClient.top );
   SetProp( hwndNew, PROP_WIDTH, rcClient.right - rcClient.left );
   SetProp( hwndNew, PROP_HEIGHT, rcClient.bottom - rcClient.top );
   /* Repositioniere das neue Fenster */
   GetClientRect( GetParent( hwndNew ), &rcClient );
   lStyle = GetWindowLong( hwndNew, GWL_STYLE );
   AdjustWindowRect( &rcClient, lStyle, FALSE );
   SetWindowPos( hwndNew,
      ( HWND ) NULL,
      rcClient.left, rcClient.top,
      rcClient.right - rcClient.left, rcClient.bottom - rcClient.top,
      SWP_NOREDRAW | SWP_NOACTIVATE );
   /* Setze den Vollbild-Fensterstil */
   SetWindowLong( hwndNew, GWL_STYLE, lStyle | WS_MAXIMIZE );
   /* Aktualisiere Menü */
   MdiZoomMenu( hwndNew );
   /* Zeige das neue Fenster */
   ShowWindow( hwndNew, SW_SHOW );
   /* Der Zeichenbereich ist der des alten Fensters */
   InvalidateRect( hwndNew, ( LPRECT ) NULL, TRUE );
   /* ALTES Dokument-Fenster */
   /* Setze den Fensterstil "Normale Größe" */
   lStyle = GetWindowLong( hwndCur, GWL_STYLE );
   SetWindowLong( hwndCur, GWL_STYLE, lStyle & ~WS_MAXIMIZE );
   /* Verschiebe Fenster dorthin, wo es vor der Vollbild-Darstellung war */
   SetWindowPos( hwndCur,
      ( HWND ) NULL,
      GetProp( hwndCur, PROP_LEFT ),
      GetProp( hwndCur, PROP_TOP ),
      GetProp( hwndCur, PROP_WIDTH ),
      GetProp( hwndCur, PROP_HEIGHT ),
      SWP_NOACTIVATE | SWP_NOZORDER );
   /* Aktualisiere Menü */
   MdiRestoreMenu( hwndCur );
}

/*
 * MdiUnhideChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *
 * Entscheide, welches Dokument als nächstes aktiviert wird, wenn das
 * aktuelle geschlossen oder verborgen wird.
 */

HWND MdiChooseNewActiveChild(
   HWND         hwndChild )
{
   HWND         hwndIter;          /* Bezug auf Fenster beim Aufzählen */

   /* Bestimme ein neues MDI-Tochterfenster für die Aktiverung */
   for ( hwndIter = GetWindow( hwndChild, GW_HWNDFIRST );
      hwndIter != NULL;
      hwndIter = GetWindow( hwndIter, GW_HWNDNEXT ) )
   {
      /* Nimm das nächste sichtbare MDI-Tochterfenster in der internen
         Fensterliste, welches nicht das aktuelle ist */
      if ( GetProp( hwndIter, PROP_ISMDI )
         && hwndIter != hwndChild
         && IsWindowVisible( hwndIter ) )
      {
         break;
      }
   }
   return hwndIter;
}

/*
 * MdiHideChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *
 * Verberge das angegebene Dokument-Fenster. Diese Routine aktiviert auch
 * das nächste verfügbare Fenster. Wenn kein weiteres Dokumentfenster
 * existiert, wird die Original-Menüleiste des MDI-Hauptfensters gesetzt und
 * der Fokus diesem Hauptfenster zugewiesen.
 */

void MdiHideChild(
   HWND         hwndChild )
{
   int          iCount;            /* Anzahl verborgener Dokumente */
   HMENU        hmenuMain;         /* Bezug auf MDI-Hauptfenster-Menüleiste */
   HMENU        hmenuOld;          /* Bezug auf Tochterfenster-Menüleiste */
   HWND         hwndMain;          /* Bezug auf MDI-Hauptfenster */
   HWND         hwndNext;          /* Bezug auf nächstes aktives Dok.-Fenster */

   /* Ein MDI-Tochterfenster weniger ist sichtbar */
   hwndMain = GetParent( hwndChild );
   iCount = GetProp( hwndMain, PROP_HIDDEN );
   SetProp( hwndMain, PROP_HIDDEN, ++iCount );
   /* Das Fenster ist nicht länger sichtbar */
   MdiDeactivateChild( hwndChild );
   MdiRemoveWindowFromMenu( hwndChild, FALSE );
   /* Finde ein neues aktives MDI-Tochterfenster */
   hwndNext = MdiChooseNewActiveChild( hwndChild );
   /* Bringe die Fenster in die neue Anordnung */
   if ( hwndNext )
   {  /* Es wurde ein anderes Fenster zur Aktivierung gefunden */
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         MdiSwitchZoom( hwndNext, hwndChild );
      }
```

Listing 3: *(Fortsetzung)*

```
      ShowWindow( hwndChild, SW_HIDE );
      MdiActivateChild( hwndNext, FALSE );
   }
   else
   {
      /* Verberge das letzte MDI-Tochterfenster */
      if ( GetProp( hwndMain, PROP_ZOOM ) )
      {
         SetProp( hwndMain, PROP_ZOOM, FALSE );
         MdiRestoreChild( hwndChild, FALSE );
      }
      ShowWindow( hwndChild, SW_HIDE );
      /* Zeige die ursprüngliche Menüleiste (ohne "Fenster"-Menü) an */
      hmenuOld = GetMenu( hwndMain );
      hmenuMain = GetProp( hwndMain, PROP_MAINMENU );
      MdiWindowMenu( hwndMain, hmenuMain, TRUE );
      SetMenu( hwndMain, hmenuMain );
      MdiWindowMenu( hwndMain, hmenuOld, FALSE );
      SetProp( hwndMain, PROP_ACTIVE, NULL );
      /* MDI-Hauptfenster besitzt jetzt den Eingabefokus */
      SetFocus( hwndMain );
   }
}

/*
 * MdiUnhideChild( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Fenster
 *
 * Zeige ein vorher verborgenes Dokument wieder an.
 */

void MdiUnhideChild(
   HWND        hwndChild )
{
   int         iCount;           /* Anzahl der verborgenen Dokumente */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */

   /* Ein MDI-Tochterfenster mehr ist sichtbar */
   hwndMain = GetParent( hwndChild );
   iCount = GetProp( hwndMain, PROP_HIDDEN );
   SetProp( hwndMain, PROP_HIDDEN, --iCount );
   /* Passe Menüleiste an */
   MdiReinsertWindowInMenu( hwndChild );
   /* Das Fenster wird aktiviert */
   if ( GetProp( hwndMain, PROP_ZOOM ) )
   {
      MdiSwitchZoom( hwndChild, GetProp( hwndMain, PROP_ACTIVE ) );
   }
   MdiActivateChild( hwndChild, TRUE );

}

/*
 * MdiDlgUnhide( hwndUnhide, message, wParam, lParam ) : int;
 *
 *    hwndUnhide     Bezug auf Fenster des Dialogfelds
 *    message        Übergebene Nachricht
 *    wParam         WORD-Parameter der Nachricht
 *    lParam         LONG-Parameter der Nachricht
 *
 * Verarbeite die Nachrichten des Dialogfelds "Anzeigen". Die Routine geht
 * davon aus, daß die Dokument-Fenster unterschiedliche Titel-Namen besitzen.
 * Wenn der Benutzer einen Titel im Verzeichnisfeld auswählt, sucht die
 * Routine das existierende Dokument als Fenster mit dem gleichen Titel.
 * Dieses wird dann wieder dargestellt.
 */

int FAR PASCAL MdiDlgUnhide(
   HWND        hwndUnhide,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   char        sTitle[80];       /* Titel des Dokumentenfensters */
   char        sSelected[80];    /* Ausgewähltes Dokument in dem Verz.-Feld */
   int         iIndex;           /* Index innerhalb des Verzeichnisfelds */
   HWND        hwndIter;         /* Bezug auf Fenster beim Durchzählen */
   HWND        hwndMain;         /* Bezug auf MDI-Hauptfenster */

   switch ( message )
   {
   case WM_INITDIALOG:
      /* Setze den Namen der MDI-Tochterfenster in das Verzeichnisfeld */
      hwndMain = GetParent( hwndUnhide );
      for ( hwndIter = GetTopWindow( hwndMain );
         hwndIter != NULL;
         hwndIter = GetWindow( hwndIter, GW_HWNDNEXT ) )
      {
         /* Übergehe nicht-MDI oder sichtbare Fenster */
         if ( !GetProp( hwndIter, PROP_ISMDI )
            || IsWindowVisible( hwndIter ) )
         {
            continue;
         }
         /* Ein mögliches Fenstr zum Wiederanzeigen */
         GetWindowText( hwndIter, sTitle, sizeof( sTitle ) );
         SendDlgItemMessage( hwndUnhide,
            DLGUNHIDE_LB,
            LB_ADDSTRING,
            0,
            ( LONG ) ( LPSTR ) sTitle );
      }
      /* wähle das erste MDI-Tochterfenster in dem Verzeichnisfeld aus */
      SendDlgItemMessage( hwndUnhide, DLGUNHIDE_LB, LB_SETCURSEL, 0, 0L );
      return TRUE;
   case WM_COMMAND:
      switch( wParam )
      {
```

Listing 3: *(Fortsetzung)*

```
      case IDOK:
         /* welches MDI-Tochterfenster wurde im Verzeichnis ausgewählt? */
         iIndex = ( int ) SendDlgItemMessage( hwndUnhide,
            DLGUNHIDE_LB,
            LB_GETCURSEL,
            0,
            0L );
         if ( iIndex == LB_ERR )
         {
            EndDialog( hwndUnhide, NULL );
            break;
         }
         /* bestimme den Titel des ausgewählten MDI-Tochterfensters */
         SendDlgItemMessage( hwndUnhide,
            DLGUNHIDE_LB,
            LB_GETTEXT,
            iIndex,
            ( LONG ) ( LPSTR ) sSelected );
         /* vergleiche den Titel mit allen MDI-Tochterfenstern */
         hwndMain = GetParent( hwndUnhide );
         for ( hwndIter = GetTopWindow( hwndMain );
            hwndIter != NULL;
            hwndIter = GetWindow( hwndIter, GW_HWNDNEXT ) )
         {
            /* Übergehe nicht-MDI oder sichtbare Fenster */
            if ( !GetProp( hwndIter, PROP_ISMDI )
               || IsWindowVisible( hwndIter ) )
            {
               continue;
            }
            /* Eines der versteckten Fenstern wurde gefunden */
            GetWindowText( hwndIter, sTitle, sizeof( sTitle ) );
            if ( strcmp( sTitle, sSelected ) == 0 )
            {
               /* Das Fenster wurde gefunden. Dies beruht auf der Tatsache,
                  daß keine zwei MDI-Tochterfenster den gleichen Titelnamen
                  haben können. */
               EndDialog( hwndUnhide, ( int ) hwndIter );
               return FALSE;
            }
         }
         /* Kein MDI-Tochterfenster ausgewählt */
         EndDialog( hwndUnhide, NULL );
         break;
      case IDCANCEL:
         EndDialog( hwndUnhide, NULL );
         break;
      case DLGUNHIDE_LB:
         /* Doppel-Klick im Verzeichnisfeld bewirkt das gleiche wie OK */
         if ( HIWORD( lParam ) == LBN_DBLCLK )
         {
            SendMessage( hwndUnhide, WM_COMMAND, IDOK, 0L );
         }
         break;
      }
      break;
   }
   return FALSE;
}
```

Listing 3: *(Ende)*

```
/*
 * MDI3.C - Menu Handling Routines
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Dieses Modul enthält den gesamten MDI-Code zum Verwalten der Menüs. Hier
 * wird das Dokument-Steuermenü in die Menüleiste des Hauptfensters einge-
 * fügt und auch wieder entfernt. Weiterhin werden die Titel der neu erzeugten
 * oder wieder angezeigten Dokumente in das "Fenster"-Menü aufgenommen und
 * entfernt, wenn sie zerstört oder verborgen werden. Wird zwischen Dokumenten
 * umgeschaltet, wird auch die Markierung im "Fenster"-Menü umgeschaltet.
 * Schließlich wird hier das meiste der Menü-Tastatur-Schnittstelle
 * realisiert.
 *
 * Entwickelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 *
 * Deutsche Anpassungen: Marcellus Buchheit
 */

#include <STDIO.H>
#include <STRING.H>
#include <WINDOWS.H>

#include "MDI.H"
```

Listing 4: *Menüverwaltung und Tastatur-Schnittstelle*

```
/* statische Variablen */
static FARPROC  lpMsgHook;          /* für den Menü-Systemeingriff */
static FARPROC  lpOldMsgHook;

/* globale Variablen */
int             iCurrentPopup = POP_NONE;  /* Welches Menü ist angezeigt */
int             iNextPopup = POP_NONE;     /* Welches Menü ist das nächste */

/*
 * MdiZoomMenu( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Tochterfenster
 *
 * Passe das Dokument-Menüleiste an, indem das Steuermenü aufgenommen wird.
 * Dies erfolgt, wenn ein Dokument als Vollbild dargestellt wird.
 */

void MdiZoomMenu(
   HWND        hwndChild )
{
   HBITMAP     hBitmap;             /* Bezug auf Steuermenü-Rasterbild */
   HMENU       hmenuSystem;         /* Bezug auf Steuermenü */
   HWND        hwndMain;            /* Bezug auf MDI-Hauptfenster */

   /* Ändere das Menü */
   hwndMain = GetParent( hwndChild );
   hmenuSystem = MdiGetChildSysMenu( );
   hBitmap = GetProp( hwndMain, PROP_SYSMENU );
   ChangeMenu( GetProp( hwndChild, PROP_CHILDMENU ),
      0,
      ( LPSTR ) ( DWORD ) hBitmap,
      hmenuSystem,
      MF_BITMAP | MF_BYPOSITION | MF_INSERT | MF_POPUP );
}

/*
 * MdiRestoreMenu( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Tochterfensters
 *
 * Passe die Dokument-Menüleiste an, indem das MDI-Steuermenü entfernt wird.
 * Dies erfolgt, wenn ein Vollbild-Dokument wiederhergestellt wird.
 */

void MdiRestoreMenu(
   HWND        hwndChild )
{
   /* Ändere das Menü */
   ChangeMenu( GetProp( hwndChild, PROP_CHILDMENU ),
      0,
      ( LPSTR ) NULL,
      0,
      MF_BYPOSITION | MF_REMOVE );
}

/*
 * MdiAppendWindowMenu( hwndChild ) : void;
 *
 *    hwndChild      Handle to document
 *
 * Füge den Titel des neu angelegten Dokuments in das "Fenster"-Menü ein.
 * Der Index wird berechnet, indem die Anzahl der Einträge des Menüs
 * gezählt wird.
 */

void MdiAppendWindowToMenu(
   HWND        hwndChild )
{
   int         iIndex;              /* numerischer Index für "Fenster"-Menü */
   char        szCurrent[50];       /* Text des Titels */
   char        szText[50];          /* Text für Menüeintrag */
   HMENU       hmenuChild;          /* Die Menüleiste dieses Dokuments */
   HMENU       hmenuWindow;         /* "Fenster"-Menü */
   HWND        hwndMain;            /* Bezug auf MDI-Hauptfenster */

   /* Ermittele wichtige Daten */
   hwndMain = GetParent( hwndChild );
   hmenuChild = GetProp( hwndChild, PROP_CHILDMENU );
   hmenuWindow = GetProp( hwndMain, PROP_WINDOWMENU );
   /* Füge falls erforderlich Separator-Linie ein */
   if ( GetMenuItemCount( hmenuWindow ) == WINDOW_POS - 1 )
   {
      ChangeMenu( hmenuWindow,
         0,
         NULL,
         0,
         MF_APPEND | MF_SEPARATOR );
   }
   /* Jetzt Eintrag im "Windows"-Menü anhängen */
   iIndex = GetMenuItemCount( hmenuWindow ) - WINDOW_POS + 1;
   GetWindowText( hwndChild, szCurrent, sizeof( szCurrent ) );
   sprintf( szText, "&%d. %s", iIndex, szCurrent );
   ChangeMenu( hmenuWindow,
      0,
      szText,
      GetProp( hwndChild, PROP_MENUID ),
      MF_APPEND | MF_BYPOSITION | MF_CHECKED | MF_STRING );
}


/*
 * MdiReinsertWindowInMenu( hwndChild ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Tochterfenster
 *
 * Füge den Titel eines verborgenen Dokuments im "Fenster"-Menü wieder ein.
 * Der Index wird berechnet, indem die Anzahl der Einträge des Menüs
 * gezählt wird.
 */
```

Listing 4: *(Fortsetzung)*

```
void MdiReinsertWindowInMenu(
   HWND        hwndChild )
{
   char        szCurrent[50];       /* Text des Titels */
   char        szText[50];          /* Text für Menüeintrag */
   int         iIndex;              /* numerischer Index für "Fenster"-Menü */
   HMENU       hmenuWindow;         /* "Fenster"-Menü */
   HWND        hwndMain;            /* Bezug auf MDI-Hauptfenster */
   /* Ermittele wichtige Daten */
   hwndMain = GetParent( hwndChild );
   hmenuWindow = GetProp( hwndMain, PROP_WINDOWMENU );
   /* Füge falls erforderlich Separator-Linie ein */
   if ( GetMenuItemCount( hmenuWindow ) == WINDOW_POS - 1 )
   {
      ChangeMenu( hmenuWindow,
         0,
         NULL,
         0,
         MF_APPEND | MF_SEPARATOR );
   }
   /* Jetzt Eintrag im "Windows"-Menü anhängen */
   iIndex = GetMenuItemCount( hmenuWindow ) - WINDOW_POS + 1;
   GetWindowText( hwndChild, szCurrent, sizeof( szCurrent ));
   szCurrent[sizeof( szCurrent ) - 1] = '\0';
   sprintf( szText, "&%d. %s", iIndex, szCurrent );
   /* Hänge am Ende an */
   ChangeMenu( hmenuWindow,
      0,
      szText,
      GetProp( hwndChild, PROP_MENUID ),
      MF_APPEND | MF_CHECKED | MF_STRING );
   /* Zeige Änderungen an */
   DrawMenuBar( hwndMain );
}

/*
 * MdiRemoveWindowFromMenu( hwndChild, bWindowDying ) : void;
 *
 *    hwndChild      Bezug auf Dokument-Tochterfenster
 *    bWindowDying   ist TRUE wenn das Fenster zerstört wird.
 *
 * Entferne den Titel eines Dokuments aus dem "Fenster"-Menü, wenn das
 * Dokument verborgen oder zerstört wird. Wenn der Dokument-Titel nicht
 * der letzte Eintrag im Menü ist, werden die folgenden Dokument-Indizes
 * aktualisiert.
 */

void MdiRemoveWindowFromMenu(
   HWND        hwndChild,
   BOOL        bWindowDying )
{
   char        szCurrent[50];       /* Text des Titels */
   char        szText[50];          /* Text für Menüeintrag */
   int         iIndex;              /* numerischer Index für "Fenster"-Menü */
   HMENU       hmenuChild;          /* Die Menüleiste dieses Dokuments */
   HMENU       hmenuWindow;         /* "Fenster"-Menü */
   HWND        hwndMain;            /* Bezug auf MDI-Hauptfenster */

   /* Ermittele wichtige Daten */
   hwndMain = GetParent( hwndChild );
   hmenuChild = GetProp( hwndChild, PROP_CHILDMENU );
   hmenuWindow = GetProp( hwndMain, PROP_WINDOWMENU );
   /* Berechne den Index */
   for ( iIndex = WINDOW_POS;
      iIndex < GetMenuItemCount( hmenuWindow );
      iIndex++ )
   {
      if ( GetMenuItemID( hmenuWindow, iIndex )
         == GetProp( hwndChild, PROP_MENUID ) )
      {  /* Eintrag wurde gefunden */
         break;
      }
   }
   /* Entferne Eintrag im Menü */
   ChangeMenu( hmenuWindow,
      GetProp( hwndChild, PROP_MENUID ),
      NULL,
      NULL,
      MF_BYCOMMAND | MF_DELETE );
   /* Korrigiere den Rest des Menüs */
   if ( GetMenuItemCount( hmenuWindow ) == WINDOW_POS )
   {  /* Entferne Separator-Linie */
      ChangeMenu( hmenuWindow,
         WINDOW_POS - 1,
         NULL,
         0,
         MF_DELETE | MF_BYPOSITION );
   }
   else
   {  /* Verschiebe nachfolgenden Menüeinträge nach oben */
      for ( ; iIndex < GetMenuItemCount( hmenuWindow ); iIndex++ )
      {
         GetMenuString( hmenuWindow,
            iIndex,
            szCurrent,
            sizeof( szCurrent ),
            MF_BYPOSITION );
         sprintf( szText,
            "&%d. %s",
            iIndex - WINDOW_POS + 1,
            strchr( szCurrent, ' ' ) + 1 );
         ChangeMenu( hmenuWindow,
            iIndex,
            szText,
            GetMenuItemID( hmenuWindow, iIndex ),
            MF_BYPOSITION | MF_CHANGE | MF_STRING | MF_UNCHECKED );
      }
   }
}
```

Listing 4: *(Fortsetzung)*

```
/*
 * MdiWindowMenu( hwndMain, mmenuChild, bAttach ) : void
 *
 *    hwndMain        Bezug auf MDI-Hauptfenster
 *    hmenuChild      Aktuelle Dokument-Menüleiste
 *    bAttach         Füge ein oder entferne
 *
 * Füge das "Fenster"-Menü in die angegebene Dokument-Menüleiste ein oder
 * entferne das Menü.
 */

void MdiWindowMenu(
   HWND        hwndMain,
   HMENU       hmenuChild,
   BOOL        bAttach )
{
   HMENU       hmenuWindow;          /* Bezug auf "Fenster"-Menü */

   hmenuWindow = GetProp( hwndMain, PROP_WINDOWMENU );
   switch( bAttach )
   {
   case TRUE:
      /* Füge hinzu */
      ChangeMenu( hmenuChild,
         0,
         "&Fenster",
         hmenuWindow,
         MF_APPEND | MF_BYPOSITION | MF_POPUP );
      break;
   case FALSE:
      /* Entferne Menü */
      ChangeMenu( hmenuChild,
         GetMenuItemCount( hmenuChild ) - 1,
         ( LPSTR ) NULL,
         0,
         MF_BYPOSITION | MF_REMOVE );
      break;
   }
   return;
}

/*
 * MdiInitSystemMenu( hwndChild ) : void
 *
 *    hwndChild       Bezug auf Dokument-Tochterfenster
 *
 * Stelle sicher, daß das MDI-Tochterfenster-Steuermenü korrekt ist.
 * Dies ist kein Thema, wenn das Tochterfenster ein Vollbild ist, da
 * dann das Systemmenü (ein Untermenü der Menüleiste) selbst erzeugt
 * wurde. Aber wenn das Steuermenü sich in der linken oberen Ecke des
 * Tochterfensters befindet, ist es ein Standard-Fenster-Steuermenü und
 * enthält die <Alt>-Abkürzungstasten anstatt den <Strg>-Abkürzungstasten.
 */

void MdiInitSystemMenu(
   HWND        hwndChild )
{
   HMENU       hmenuSystem;       /* Bezug auf Steuermenü */

   /* Lese eine Kopie des Steuermenüs */
   hmenuSystem = GetSystemMenu( hwndChild, FALSE );
   /* Initialisiere Steuermenü */
   ChangeMenu( hmenuSystem,
      SC_RESTORE,
      "&Wiederherstellen\tStrg+F5",
      SC_RESTORE,
      GetProp( GetParent( hwndChild ), PROP_ZOOM )
         ? MF_BYCOMMAND | MF_CHANGE | MF_STRING
         : MF_BYCOMMAND | MF_CHANGE | MF_GRAYED | MF_STRING );
   ChangeMenu( hmenuSystem,
      SC_MOVE,
      "&Bewegen\tStrg+F7",
      SC_MOVE,
      MF_BYCOMMAND | MF_CHANGE | MF_STRING );
   ChangeMenu( hmenuSystem,
      SC_SIZE,
      "&Größe ändern\tStrg+F8",
      SC_SIZE,
      MF_BYCOMMAND | MF_CHANGE | MF_STRING );
   ChangeMenu( hmenuSystem,
      SC_MINIMIZE,
      ( LPSTR ) NULL,
      SC_MINIMIZE,
      MF_BYCOMMAND | MF_DELETE );
   ChangeMenu( hmenuSystem,
      SC_MAXIMIZE,
      "&Vollbild\tStrg+F10",
      SC_MAXIMIZE,
      MF_BYCOMMAND | MF_CHANGE | MF_STRING );
   ChangeMenu( hmenuSystem,
      SC_CLOSE,
      "&Schließen\tStrg+F4",
      SC_CLOSE,
      MF_BYCOMMAND | MF_CHANGE | MF_STRING );
}

/*
 * MdiGetChildSysMenu() : HMENU;
 *
 * Erzeuge das Steuermenü für ein Dokument. Das Menü wird nur dann ver-
 * wendet, wenn das Dokument als Vollbild dargestellt ist. Andernfalls
 * wird das Standard-Steuermenü verwendet, dessen Text beim Erhalt der
 * WM_INITMENU-Nachricht modifiziert wird.
 */

HMENU MdiGetChildSysMenu( void )
{
   HMENU       hmenuSystem;       /* Bezug auf Steuermenü */
```

Listing 4: *(Fortsetzung)*

```
   /* Erzeuge das Menü */
   hmenuSystem = CreateMenu( );
   ChangeMenu( hmenuSystem, 0,
      "&Wiederherstellen\tStrg+F5",
      IDM_RESTORE,
      MF_APPEND | MF_STRING );
   ChangeMenu( hmenuSystem, 0,
      "&Bewegen\tStrg+F7",
      IDM_MOVE,
      MF_APPEND | MF_GRAYED | MF_STRING );
   ChangeMenu( hmenuSystem, 0,
      "&Größe ändern\tStrg+F8",
      IDM_SIZE,
      MF_APPEND | MF_GRAYED | MF_STRING );
   ChangeMenu( hmenuSystem, 0,
      "&Vollbild\tStrg+F10",
      IDM_MAXIMIZE,
      MF_APPEND | MF_STRING );
   ChangeMenu( hmenuSystem, 0,
      ( LPSTR ) NULL,
      0,
      MF_APPEND | MF_SEPARATOR );
   ChangeMenu( hmenuSystem, 0,
      "&Schließen\tStrg+F4",
      IDM_CLOSE,
      MF_APPEND | MF_STRING );
   return hmenuSystem;
}

/*
 * MdiCreateChildSysBitmap( hwndMain ) : HBITMAP;
 *
 *    hwndMain        Handle to MDI desktop
 *
 * Erzeuge ein Rasterbild für das Steuermenü eines Dokuments. Das Raster-
 * bild ähnelt einem Minuszeichen. Es wird vom Windows-System zur Verfü-
 * gung gestellt, sodaß kein eigenes angelegt werden muß. Das System-Bild
 * besteht jedoch aus zwei Hälften: eine rechte für ein Applikationsfenster
 * (Hauptfenster) und eine linke für ein Tochterfenster. Diese Funktion
 * erzeugt einen Speicher-Zeichenbereich für das Doppel-Rasterbild und einen
 * weiteren (kleineren) für die Tochterfenster-Hälfte. Anschließend wird mit
 * BitBlt() die Tochterfensterhälfte in den zweiten Bereich übertragen und
 * dieser als Rasterbild zurückgegeben.
 */

HBITMAP MdiCreateChildSysBitmap(
   HWND        hwndMain )
{
   BOOL        bResult;         /* Haben wir ein gültiges Rasterbild? */
   HDC         hDC;             /* Zeichenbereich für MDI-Hauptfenster */
   HDC         hMemFullDC;      /* Speicher-Zeichenbereich für beide Bilder */
   HDC         hMemHalfDC;      /* Speicher-Zeichenber. für Tochterfenst.-Bild */
   BITMAP      Bitmap;          /* Rasterbild-Parameter */
   HBITMAP     hFullBitmap;     /* Bezug auf Rasterbild der beiden Bilder */
   HBITMAP     hHalfBitmap;     /* Bezug auf Rasterbild des Tochterfenst.-Bilds */
   int         iBitmapWidth;    /* Breite des System-Rasterbilds */
   int         iBitmapHeight;   /* Höhe des System-Rasterbilds */
   bResult = FALSE;
   hFullBitmap = LoadBitmap( NULL, MAKEINTRESOURCE( OBM_CLOSE ) );
   if ( hFullBitmap )
   {
      GetObject( hFullBitmap, sizeof( Bitmap ), ( LPSTR ) &Bitmap );
      iBitmapWidth = Bitmap.bmWidth / 2;
      iBitmapHeight = GetSystemMetrics( SM_CYMENU );
      hDC = GetDC( hwndMain );
      if ( hDC )
      {
         hMemFullDC = CreateCompatibleDC( hDC );
         if ( hMemFullDC )
         {
            SelectObject( hMemFullDC, hFullBitmap );
            hMemHalfDC = CreateCompatibleDC( hDC );
            if ( hMemHalfDC )
            {
               hHalfBitmap = CreateCompatibleBitmap( hMemHalfDC,
                  iBitmapWidth, iBitmapHeight );
               if ( hHalfBitmap )
               {
                  SelectObject( hMemHalfDC, hHalfBitmap );
                  PatBlt( hMemHalfDC,
                     0, 0,
                     iBitmapWidth, iBitmapHeight,
                     WHITENESS );
                  bResult = BitBlt( hMemHalfDC,
                     0, 0,
                     iBitmapWidth, iBitmapHeight,
                     hMemFullDC,
                     iBitmapWidth, 0,
                     SRCCOPY );
               }
               DeleteDC( hMemHalfDC );
            }
            DeleteDC( hMemFullDC );
         }
         ReleaseDC( hwndMain, hDC );
      }
      DeleteObject( hFullBitmap );
   }
   return bResult ? hHalfBitmap : NULL;
}

/*
 * MdiSetMenuKeyhook( hInst ) : void;
 *    hInst           Bezug auf aktuelle Instanz
 *
 * Installiere den taskorientierten Systemnachrichten-Systemeingriff zur
 * Überwachung der Tastatur-Eingänge. Damit ist es möglich, bestimmten Tasten
 * Menüeinträgen zuzuweisen.
 */
```

Listing 4: *(Fortsetzung)*

```
void MdiSetMenuKeyHook(
   HANDLE      hInst )
{
   lpMsgHook = MakeProcInstance( ( FARPROC ) MdiMsgHook, hInst );
   lpOldMsgHook = SetWindowsHook( WH_MSGFILTER, lpMsgHook );
}

/*
 * MdiMenuMessageLoopUpdate( hwndMain ) : void;
 *
 *    hwndMain       Bezug auf MDI-Hauptfenster
 *
 *
 * Aktiviere das richtige (Aufklapp-) Menü, wenn die linke oder rechte
 * Richtungstaste gedrückt wurde und das Steuermenüs eines Dokuments
 * vorhanden ist. Zu beachten ist, daß dies erst erfolgt, nachdem
 * die Funktion DispatchMessage() aufgerufen wurde. Dadurch wird erreicht,
 * daß erst das augenblicklich aufgeklappte Menü geschlossen wird,
 * bevor das nächste aufgeklappt wird.
 */

void MdiMenuMessageLoopUpdate(
   HWND        hwndMain )
{
   int         iOldNext;         /* rette augenblicklicher Status */
   HWND        hwndActive;       /* Bezug auf aktives Dokument-Fenster */

   /* ohne Nachrichten ist mit Sicherheit kein Menü aufgeklappt */
   iCurrentPopup = POP_NONE;
   if ( iNextPopup != POP_NONE )
   {
      /* diese Sequenz ist nötig, weil der SendMessage()-Aufruf reentrant-
         Probleme erzeugen kann */
      iOldNext = iNextPopup;
      iNextPopup = POP_NONE;
      /* aktiviere das korrekte Menü */
      hwndActive = GetProp( hwndMain, PROP_ACTIVE );
      switch( iOldNext )
      {
      case POP_MAINSYS:
         SendMessage( hwndMain,
            WM_SYSCOMMAND,
            SC_KEYMENU,
            (DWORD) ' ');
         break;
      case POP_MAIN1ST:
         /* wechsele zum ersten Menü in der Menüleiste ("Datei") */
         SendMessage( hwndMain,
            WM_SYSCOMMAND,
            SC_KEYMENU,
            (DWORD) 'D');
         break;
      case POP_CHILDSYS:
         if ( hwndActive )
         {
            SendMessage(
               hwndActive,
               WM_SYSCOMMAND,
               SC_KEYMENU,
               (DWORD) '-');
         }
         break;
      }
   }
}


/*
 * MdiMsgHook( iContext, wCode, lParam ) : LONG;
 *
 *    iContext       das Ziel der System-Nachricht (z.B. Menü)
 *    wCode          unbenützt
 *    lParm          Zeiger auf die eingegangene Nachricht
 *
 * Wartet auf linke und rechte Richtungstasten, wenn ein Menü aufgeklappt
 * ist. Dies ermöglicht eine Wechsel zwischen dem Steuermenü des MDI-
 * Hauptfensters, dem Dokument-Steuermenü und dem "Datei"-Menü. Zu
 * beachten ist, daß die Funktion nur globale Variablen setzt und keine
 * Aktionen ausführt.
 */

LONG FAR PASCAL MdiMsgHook(
int         iContext,
WORD        wCode,
LONG        lParam )
{
   BOOL        bCancelMenu = FALSE;
   BOOL        bActive;
   BOOL        bZoomed;
   HWND        hwndMain;
   WORD        wKey;
   MSG FAR     *lpMsg;

   lpMsg = ( MSG FAR * ) lParam;
   hwndMain = lpMsg->hwnd;
   if ( GetProp( hwndMain, PROP_ISMDI ) )
   {
      hwndMain = GetParent( hwndMain );
   }
   wKey = lpMsg->wParam;
   if ( iContext == MSGF_MENU
      && wCode == HC_ACTION
      && GetProp( hwndMain, PROP_ACTIVE )
      && !GetProp( hwndMain, PROP_ZOOM )
      && lpMsg->message == WM_KEYDOWN
      && wKey == VK_LEFT || wKey == VK_RIGHT )
   {
      /* Linke oder rechte Richtungstaste im Menü */
      switch( iCurrentPopup )
```

Listing 4: *(Fortsetzung)*

```
      {
      case POP_CHILDSYS:
         iNextPopup = ( wKey == VK_LEFT ? POP_MAINSYS : POP_MAIN1ST );
         bCancelMenu = TRUE;
         break;
      case POP_MAINSYS:
         if ( wKey == VK_RIGHT )
         {
            iNextPopup = POP_CHILDSYS;
            bCancelMenu = TRUE;
         }
         break;
      case POP_MAIN1ST:
         if ( wKey == VK_LEFT )
         {
            iNextPopup = POP_CHILDSYS;
            bCancelMenu = TRUE;
         }
         break;
      }
      /* Beende dieses Menü */
      if ( bCancelMenu )
      {
         lpMsg->wParam = VK_CANCEL;
      }
   }
   /* Ende des Systemeingriffs */
   return DefHookProc( iContext,
      wCode,
      lParam,
      ( FARPROC FAR * ) &lpOldMsgHook );
}

/*
 * MdiFreeMenuKeyhook( void ) : void
 *
 * Gibt die Prozedurinstanz frei, die für den Menü-Systemeingriff
 * benötigt wurde.
 */

void MdiFreeMenuKeyHook( void )
{
   FreeProcInstance( lpMsgHook );
}
```

Listing 4: *(Ende)*

```
# Flags für Compilieren
CFLAGS=-AM -c -Gsw -Osal -W2 -Zp

# MDI
mdi1.obj: mdi1.c mdi.h
   cl $(CFLAGS) mdi1.c

mdi2.obj: mdi2.c mdi.h
   cl $(CFLAGS) mdi2.c

mdi3.obj: mdi3.c mdi.h
   cl $(CFLAGS) mdi3.c

mdi.lib: mdi1.obj mdi2.obj mdi3.obj
   lib mdi-+mdi1-+mdi2-+mdi3;
   del mdi.bak
```

Listing 5: *Die MAKE-Datei MDI zum Erzeugen der MDI-Bibliothek MDI.LIB*

```
# Flags für Compilieren
CFLAGS=-AM -c -Gsw -Osal -W2 -Zp

# COLORS
colors.res: colors.rc colors.ico colors.h mdi.h
   rc -r colors.rc

colors.obj: colors.c colors.h mdi.h
   cl $(CFLAGS) colors.c

colors.exe: colors.obj colors.def mdi.lib
   link4 colors,colors/ALIGN:16,colors,mdi+mlibw+mlibcew/NOE,colors
   rc colors.res

colors.exe: colors.res
   rc colors.res
```

Listing 6: *Die MAKE-Datei COLORS zum Erzeugen der MDI-Beispielapplikation*

```
NAME            COLORS
DESCRIPTION     'Mehrfach-Dokumenten-Schnittstelle'
STUB                    'WINSTUB.EXE'

CODE            MOVEABLE DISCARDABLE PURE LOADONCALL SHARED
DATA            MOVEABLE MULTIPLE

HEAPSIZE        2048
STACKSIZE       2048

EXPORTS
   MainWndProc  @1
   ColorWndProc @2
   MainDlgNew   @3
   MainDlgAbout @4
   MdiMsgHook   @5
   MdiDlgUnhide @6
```

Listing 7: *Die Linker-Definitionsdatei COLORS.DEF*

```
/*
 * COLORS.RC - Ressourcen-Datei für Applikation COLORS
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Entwicklelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 *
 * Deutsche Anpassungen: Marcellus Buchheit
 */

/*
 * Spezifikation der Kopfdateien für COLORS
 */
#include <WINDOWS.H>
#include "COLORS.H"
#include "MDI.H"

MainIcon ICON colors.ico

MainMenu MENU
BEGIN
   POPUP "&Datei"
   BEGIN
      MENUITEM "&Neu...",               IDM_NEW
      MENUITEM "&Öffnen...",            IDM_OPEN, GRAYED
      MENUITEM "&Schließen",            IDM_CLOSE, GRAYED
      MENUITEM "S&peichern",            IDM_SAVE, GRAYED
      MENUITEM "Speichern &unter...", IDM_SAVEAS, GRAYED
      MENUITEM SEPARATOR
      MENUITEM "&Beenden",              IDM_EXIT
      MENUITEM "&Über COLORS...",       IDM_ABOUT
   END
END

RedMenu MENU
BEGIN
   POPUP "&Datei"
   BEGIN
      MENUITEM "&Neu...",               IDM_NEW
      MENUITEM "&Öffnen...",            IDM_OPEN, GRAYED
      MENUITEM "&Schließen",            IDM_CLOSE
      MENUITEM "S&peichern",            IDM_SAVE, GRAYED
      MENUITEM "Speichern &unter...", IDM_SAVEAS, GRAYED
      MENUITEM SEPARATOR
      MENUITEM "&Beenden",              IDM_EXIT
      MENUITEM "&Über COLORS...",       IDM_ABOUT
   END
   POPUP "&Rot"
   BEGIN
      MENUITEM "&0 %",                  IDM_0
      MENUITEM "&25 %",                 IDM_25
      MENUITEM "&50 %",                 IDM_50
      MENUITEM "&75 %",                 IDM_75
      MENUITEM "&100 %",                IDM_100, CHECKED
   END
END

GreenMenu MENU
BEGIN
   POPUP "&Datei"
   BEGIN
      MENUITEM "&Neu...",               IDM_NEW
      MENUITEM "&Öffnen...",            IDM_OPEN, GRAYED
      MENUITEM "&Schließen",            IDM_CLOSE
      MENUITEM "S&peichern",            IDM_SAVE, GRAYED
      MENUITEM "Speichern &unter...", IDM_SAVEAS, GRAYED
      MENUITEM SEPARATOR
```

Listing 8: *Die Kopfdatei COLORS.RC der MDI-Beispielapplikation*

```
      MENUITEM "&Beenden",              IDM_EXIT
      MENUITEM "&Über COLORS...",       IDM_ABOUT
   END
   POPUP "&Grün"
   BEGIN
      MENUITEM "&0 %",                  IDM_0
      MENUITEM "&25 %",                 IDM_25
      MENUITEM "&50 %",                 IDM_50
      MENUITEM "&75 %",                 IDM_75
      MENUITEM "&100 %",                IDM_100, CHECKED
   END
END

BlueMenu MENU
BEGIN
   POPUP "&Datei"
   BEGIN
      MENUITEM "&Neu...",               IDM_NEW
      MENUITEM "&Öffnen...",            IDM_OPEN, GRAYED
      MENUITEM "&Schließen",            IDM_CLOSE
      MENUITEM "S&peichern",            IDM_SAVE, GRAYED
      MENUITEM "Speichern &unter...", IDM_SAVEAS, GRAYED
      MENUITEM SEPARATOR
      MENUITEM "&Beenden",              IDM_EXIT
      MENUITEM "&Über COLORS...",       IDM_ABOUT
   END
   POPUP "&Blau"
   BEGIN
      MENUITEM "&0 %\t  0",             IDM_0
      MENUITEM "&25 %\t  2",            IDM_25
      MENUITEM "&50 %\t  5",            IDM_50
      MENUITEM "&75 %\t  7",            IDM_75
      MENUITEM "&100 %\t  1",           IDM_100, CHECKED
   END
END

BlueAccel   ACCELERATORS
BEGIN
   "0",         IDM_0
   "2",         IDM_25
   "5",         IDM_50
   "7",         IDM_75
   "1",         IDM_100
END

STRINGTABLE
BEGIN
   IDS_TITLE,              "Mehrfach-Dokument-Schnittstelle (MDI)"
   IDS_MAINCLASS,          "MdiMainClass"
   IDS_COLORCLASS,         "MdiChildClass"
END

MainNew  DIALOG  50,50,162,60
STYLE WS_DLGFRAME|WS_POPUP
BEGIN
   GROUPBOX "Neu",                   -1,   4,  4, 100, 52
   RADIOBUTTON "&Rot",      DLGNEW_RED,   8, 16,  88, 10
   RADIOBUTTON "&Grün",     DLGNEW_GREEN, 8, 28,  88, 10
   RADIOBUTTON "&Blau",     DLGNEW_BLUE,  8, 40,  88, 10
   DEFPUSHBUTTON "OK",            IDOK, 108,  8,  50, 14
   PUSHBUTTON "Abbrechen",   IDCANCEL, 108, 28,  50, 14
END

MainAbout  DIALOG  22, 17, 156, 100
STYLE WS_DLGFRAME|WS_POPUP
BEGIN
    CTEXT "Mehrfach-Dokumenten-Schnittstelle",  -1,  0,  8,152,  8
    CTEXT "Von Geoffrey D. Nicholls",           -1,  0, 20,152,  8
    CTEXT "und Kevin P. Welch",                 -1,  0, 28,152,  8
    CTEXT "(C) Copyright 1988",                 -1,  0, 40,152,  8
    CTEXT "Eikon Systems, Inc.",                -1,  0, 48,152,  8
    CTEXT "Version 1.00  1-Nov-88",             -1,  0, 60,152,  8
    DEFPUSHBUTTON "Ok",                       IDOK, 60, 80, 32, 14, WS_GROUP
END

/*
 * MDI-Teil der Ressourcen-Datei
 */
MdiMenu MENU
BEGIN
   MENUITEM "&Neu",                      IDM_NEWWINDOW, GRAYED
   MENUITEM SEPARATOR
   MENUITEM "N&ebeneinander",            IDM_ARRANGE, GRAYED
   MENUITEM "&Immer Nebeneinander",      IDM_ARRANGEALL, GRAYED
   MENUITEM SEPARATOR
   MENUITEM "&Verbergen",                IDM_HIDE, GRAYED
   MENUITEM "&Anzeigen...",              IDM_UNHIDE
END
MdiChildAccel  ACCELERATORS
BEGIN
   VK_F4,       IDM_CLOSE,        VIRTKEY, NOINVERT, CONTROL
   VK_F5,       IDM_RESTORE,      VIRTKEY, NOINVERT, CONTROL
   VK_F6,       IDM_NEXTWINDOW,   VIRTKEY, NOINVERT, CONTROL
   VK_F6,       IDM_PREVWINDOW,   VIRTKEY, NOINVERT, CONTROL, SHIFT
   VK_F7,       IDM_MOVE,         VIRTKEY, NOINVERT, CONTROL
   VK_F8,       IDM_SIZE,         VIRTKEY, NOINVERT, CONTROL
   VK_F10,      IDM_MAXIMIZE,     VIRTKEY, NOINVERT, CONTROL
END

MdiUnhide  DIALOG 50, 50, 150, 68
STYLE WS_DLGFRAME|WS_POPUP
BEGIN
   LTEXT "&Anzeigen",                        -1,   4,  4,  88, 10
   LISTBOX                         DLGUNHIDE_LB,   4, 16,  88, 48, WS_TABSTOP
   DEFPUSHBUTTON "OK"                      IDOK,  96,  8,  50, 14
   PUSHBUTTON "Abbrechen"              IDCANCEL,  96, 28,  50, 14
END
```

Listing 8: *COLORS.RC (Ende)*

```
/*
 * COLORS.H - Kopfdatei für Applikation COLORS
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Entwicklelt von:
 *   Geoffrey Nicholls
 *   Kevin P. Welch
 *
 * (C) Copyright 1988
 * Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260
 * Foster City  CA  94404
 */

/*
 * Ressourcen-Datei-Konstanten
 */

/* Zeichenketten */
#define IDS_TITLE       1
#define IDS_MAINCLASS   2
#define IDS_COLORCLASS 3

/* Menüeintrag für Fehlersuche */
#define IDM_DEBUG       0x100

/* Einträge im Menü "Datei" */
#define IDM_NEW         0x101
#define IDM_OPEN        0x102
#define IDM_SAVE        0x103
#define IDM_SAVEAS      0x104
#define IDM_PRINT       0x105
#define IDM_ABOUT       0x106
#define IDM_EXIT        0x107

/* Einträge im Menü "Farben" */
#define IDM_0           0x108
#define IDM_25          0x109
#define IDM_50          0x10a
#define IDM_75          0x10b
#define IDM_100         0x10c

/* Einträge im Dialogfeld "Neu" */
#define DLGNEW_RED      0x100
#define DLGNEW_GREEN    0x101
#define DLGNEW_BLUE     0x102

/*
 * Weitere Fenster-Konstanten
 */

#define WE_COLOR        0
#define WE_SHADE        2
#define WE_EXTRA        4

#define COLOR_RED       0
#define COLOR_GREEN     1
#define COLOR_BLUE      2

/*
 * Prototypen der Funktionen
 */

int PASCAL       WinMain( HANDLE, HANDLE, LPSTR, int );
HWND             MainInit( HANDLE, HANDLE, int );
long FAR PASCAL MainWndProc( HWND, unsigned, WORD, LONG );
BOOL             ColorInit( HANDLE );
HWND             ColorCreate( HWND, int );
long FAR PASCAL ColorWndProc( HWND, unsigned, WORD, LONG );
int FAR PASCAL  MainDlgNew( HWND, unsigned, WORD, LONG );
int FAR PASCAL  MainDlgAbout( HWND, unsigned, WORD, LONG );
```

Listing 9: *COLORS.H, Kopfdatei des MDI-Beispiels*

```
/*
 * COLORS.C - Farbige MDI-Tochterfenster
 *
 * Programmiersprache:   Microsoft-C 5.1
 * Speichermodell:       medium
 * Entwicklungsumgebung: Microsoft Windows 2.1 SDK
 * Status:               einsatzfähig
 *
 * Dieses Modul enthält den gesamten applikationsspezifischen Code für die
 * Applikation COLORS. Diese besitzt eine MDI-Oberfläche in der sich eine
 * beliebige Anzahl von farbigen "Dokumenten" befinden kann. Diese sind
 * entweder rot, grün oder blau und können in 25%-Schritten mit Grauwerten
 * (von 0 bis 100%) versehen werden. Um die Technik der Abkürzungstasten
 * zu demonstrieren besitzt nur ein Fenster mit einem "blauen Dokument"
 * Abkürzungstasten.
 *
 * Entwickelt von: Geoffrey Nicholls, Kevin P. Welch
 *
 * (C) Copyright 1988 Eikon Systems, Inc.
 * 989 E. Hillsdale Blvd, Suite 260, Foster City  CA  94404
 */
```

Listing 10: *COLORS.C: Beispiel einer MDI-Applikation*

```
#include <STRING.H>
#include <STDIO.H>
#include <WINDOWS.H>

#include "COLORS.H"
#include "MDI.H"

/*
 * Statische Variablen
 */

/* Text für Fenster-Abbildungsbereich */
static char     *szShadings[5] = {"0 %", "25 %", "50 %", "75 %", "100 %"};

/* Titel der Dokumente */
static char     *szTitles[3] = { "Rot", "Grün", "Blau" };

/* Zähler der einzelnen Dokumente (für die Titel) */
static int      wCounts[3] = { 0, 0, 0 };

/* Farben- und Graustufen-Tabelle */
static DWORD    rgbColors[3][5] = {
                  {  /* RED */
                  RGB(255,255,255),    /*   0 % */
                  RGB(255,192,192),    /*  25 % */
                  RGB(255,128,128),    /*  50 % */
                  RGB(255,64,64),      /*  75 % */
                  RGB(255,0,0)         /* 100 % */
                  },
                  {  /* GREEN */
                  RGB(255,255,255),    /*   0 % */
                  RGB(192,255,192),    /*  25 % */
                  RGB(128,255,128),    /*  50 % */
                  RGB(64,255,64),      /*  75 % */
                  RGB(0,255,0)         /* 100 % */
                  },
                  {  /* BLUE */
                  RGB(255,255,255),    /*   0 % */
                  RGB(192,192,255),    /*  25 % */
                  RGB(128,128,255),    /*  50 % */
                  RGB(64,64,255),      /*  75 % */
                  RGB(0,0,255)         /* 100 % */
                  } };


/*
 * PASCAL WinMain( hPrevInst, hInst, lpszCommand, nCmdShow ) : int;
 *
 *    hPrevInst      Bezug auf vorherige Instanz
 *    hInst          Bezug auf aktuelle Instanz
 *    lpszCmdLine    Zeiger auf die Argumente der Befehlszeile
 *    nCmdShow       Parameter für ShowWindow()
 *
 * Dies ist die erste Funktion, welche das Systems aufruft.
 * Sie ruft die Initialisierung auf und enthält die Nachrichtenschleife.
 */

int PASCAL WinMain(
   HANDLE      hInst,
   HANDLE      hPrevInst,
   LPSTR       lpszCmdLine,
   int         nCmdShow )
{
   HWND        hwndColors;        /* Bezug auf MDI-Hauptfenster */
   MSG         msg;               /* Aktuelle Nachricht */

   /* Initialisiere alles, was die Applikation verwendet */
   hwndColors = MainInit( hPrevInst, hInst, nCmdShow );
   if ( !hwndColors )
   {
      /* Fehler beim Initialisieren */
      return NULL;
   }
   /* Verarbeite Nachrichten */
   while ( MdiGetMessage( hwndColors, &msg, NULL, NULL, NULL ) )
   {
      /* Standard-Nachrichten-Verarbeitung */
      if ( !MdiTranslateAccelerators( hwndColors, &msg ) )
      {
         TranslateMessage( &msg );
         DispatchMessage( &msg );
      }
   }
   /* Applikation beendet */
   return msg.wParam;
}


/*
 * MainInit( hPrevInst, hInst, nCmdShow ) : HWND;
 *
 *    hPrevInst      Bezug auf vorherige Instanz
 *    hInst          Bezug auf aktuelle Instanz
 *    nCmdShow       Parameter für ShowWindow()
 *
 * Zunächst wird die MDI-Oberfläche und die Farben-Dokumentfenster
 * initialisiert. Anschließend wird die MDI-Oberfläche erzeugt und ein
 * rotes Dokumentfenster angelegt.
 */

HWND MainInit(
   HANDLE      hPrevInst,
   HANDLE      hInst,
   int         nCmdShow )
{
   char        szTitle[80];       /* Titel des MDI-Hauptfensters */
   char        szClass[80];       /* Klassenname des MDI-Hauptfensters */
   HWND        hwndColors;        /* Bezug des MDI-Hauptfensters */
   WNDCLASS    WndClass;          /* Struktur der Fensterklasse */
```

Listing 10: *(Fortsetzung)*

```
   /* Erzeuge Fensterklassen */
   if ( !hPrevInst )
   {
      /* lese Fensterklassenname des Hauptfensters (Applikationsfenster) */
      LoadString( hInst, IDS_MAINCLASS, szClass, sizeof( szClass ) );
      /* setze Registrierungswerte */
      memset( &WndClass, 0, sizeof( WndClass ) );
      WndClass.style           = CS_HREDRAW | CS_VREDRAW;
      WndClass.lpfnWndProc     = MainWndProc;
      WndClass.hInstance       = hInst;
      WndClass.hIcon           = LoadIcon( hInst, "MainIcon" );
      WndClass.hCursor         = LoadCursor( NULL, IDC_ARROW );
      WndClass.hbrBackground   = COLOR_APPWORKSPACE + 1;
      WndClass.lpszMenuName    = "MainMenu";
      WndClass.lpszClassName   = szClass;
      /* registriere Hauptklasse */
      if ( !RegisterClass( &WndClass ) )
         return NULL;
      /* jedes der MDI-Tochterfenster hat seine eigene Initialisierung */
      if ( !ColorInit(hInst) )
         return NULL;
   }
   /* lege MDI-Hauptfenster (Applikationsfenster) an */
   LoadString( hInst, IDS_TITLE, szTitle, sizeof( szTitle ) );
   hwndColors = MdiMainCreateWindow( szClass,
      szTitle,
      WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
      CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
      NULL,
      NULL,
      hInst,
      NULL );
   /* wurde es erfolgreich angelegt? */
   if ( !hwndColors )
      return NULL;
   /* erzeuge ein rotes Tochterfenster */
   if ( !ColorCreate( hwndColors, COLOR_RED ) )
   {
      DestroyWindow( hwndColors );
      return NULL;
   }
   /* fertig: zeige alles an */
   ShowWindow( hwndColors, nCmdShow );
   UpdateWindow( hwndColors );
   /* Rückkehr */
   return hwndColors;
}


/*
 * MainWndProc( hwndMain, message, wParam, lParam ) : LONG;
 *
 *    hwndColors     Bezug des MDI-Hauptfensters
 *    message        übergebene Nachricht
 *    wParam         WORD-Parameter der Nachricht
 *    lParam         LONG-Parameter der Nachricht
 *
 * Verarbeite Nachrichten an das MDI-Hauptfenster. Dies beinhaltet auch
 * alle WM_COMMAND-Nachrichten, die übergeben werden, wenn KEIN Dokument im
 * Hauptfenster abgebildet ist.
 */


long FAR PASCAL MainWndProc(
   HWND        hwndColors,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   FARPROC     lpProc;           /* Prozedur-Instanz für Dialogfelder */
   HANDLE      hInst;            /* Bezug auf Instanz der Applikation */

   switch ( message )
   {
   case WM_COMMAND:
      hInst = GetWindowWord( hwndColors, GWW_HINSTANCE );
      switch( wParam )
      {
      case IDM_NEW:
         /* Dialogfeld "Neu" */
         lpProc = MakeProcInstance( MainDlgNew, hInst );
         switch( DialogBox( hInst, "MainNew", hwndColors, lpProc ) )
         {
         case DLGNEW_RED:
            ColorCreate( hwndColors, COLOR_RED );
            break;
         case DLGNEW_GREEN:
            ColorCreate( hwndColors, COLOR_GREEN );
            break;
         case DLGNEW_BLUE:
            ColorCreate( hwndColors, COLOR_BLUE );
            break;
         }
         FreeProcInstance( lpProc );
         break;
      case IDM_OPEN:
         break;
      case IDM_ABOUT:
         /* Dialogfeld "Über" */
         lpProc = MakeProcInstance( MainDlgAbout, hInst );
         DialogBox( hInst, "MainAbout", hwndColors, lpProc );
         FreeProcInstance( lpProc );
         break;
      case IDM_EXIT:
         /* Leite als Schließen-Befehl weiter */
         PostMessage( hwndColors, WM_SYSCOMMAND, SC_CLOSE, 0L );
         break;
      }
      break;
```

Listing 10: *(Fortsetzung)*

```
   case WM_DESTROY:
      PostQuitMessage( 0 );
      break;
   }
   return MdiMainDefWindowProc( hwndColors, message, wParam, lParam );
}

/*
 * ColorInit( hInst ) : BOOL;
 *
 *    hInst          Bezug auf Instanz der Applikation
 *
 * Registriere die Fensterklasse der Tochterfenster (Dokumente)
 */

BOOL ColorInit(
   HANDLE      hInst )
{
   char        szClass[80];      /* Name der Fensterklasse */
   WNDCLASS    WndClass;         /* Struktur der Fensterklasse */

   /* lese Fensterklassen-Namen ein */
   LoadString( hInst, IDS_COLORCLASS, szClass, sizeof( szClass ) );
   /* setze Registrierungswerte */
   memset( &WndClass, 0, sizeof( WndClass ) );
   WndClass.style           = CS_HREDRAW | CS_VREDRAW;
   WndClass.lpfnWndProc     = ColorWndProc;
   WndClass.cbWndExtra      = WE_EXTRA;
   WndClass.hInstance       = hInst;
   WndClass.hCursor         = LoadCursor( NULL, IDC_ARROW );
   WndClass.hbrBackground   = GetStockObject( GRAY_BRUSH );
   WndClass.lpszClassName   = szClass;
   /* registriere Klasse */
   return RegisterClass( &WndClass );
}

/*
 * ColorCreate( hwndParent, wType ) : HWND;
 *
 *    hwndParent     Bezug auf das MDI-Hauptfenster
 *    wType          Farbe des Dokuments
 *
 * Erzeuge ein Tochterfenster des MDI-Hauptfensters für ein Dokument der
 * angegebenen Farbe. Es wird die entsprechende Menüleiste geladen und,
 * wenn die angegebene Farbe blau ist, die Tabelle mit den Abkürzungstasten
 * geladen. In dem reservierten Fensterdatenbereich wird die Farbe und der
 * Grauwert des Dokuments initialisiert.
 */

HWND ColorCreate(
   HWND        hwndParent,
   int         wType )
{
   char        szClass[80];      /* Klassenname für Dokument */
   char        szTitle[80];      /* Titel für dieses Dokument */
   HANDLE      hAccel = NULL;    /* Abkürzungstasten nur für blau */
   HANDLE      hInst;            /* Bezug auf Applikations-Instanz */
   HMENU       hmenuChild;       /* Bezug auf die Menüleiste des Dokuments */
   HWND        hwndChild;        /* Bezug auf Fenster des Dokuments */

   /* Lese wichtige Daten */
   hInst = GetWindowWord( hwndParent, GWW_HINSTANCE );
   LoadString( hInst, IDS_COLORCLASS, szClass, sizeof( szClass ) );
   sprintf( szTitle, "%s%d", szTitles[wType], ++wCounts[wType] );
   switch( wType )
   {
   case COLOR_RED:
      hmenuChild = LoadMenu( hInst, "RedMenu" );
      break;
   case COLOR_GREEN:
      hmenuChild = LoadMenu( hInst, "GreenMenu" );
      break;
   case COLOR_BLUE:
      hmenuChild = LoadMenu( hInst, "BlueMenu" );
      hAccel = LoadAccelerators( hInst, "BlueAccel" );
      break;
   }
   /* Lege Fenster an */
   hwndChild = MdiChildCreateWindow( szClass,
      szTitle,
      WS_MDICHILD,
      CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
      hwndParent,
      hmenuChild,
      hInst,
      0L );
   /* War es erfolgreich? */
   if ( hwndChild )
   {
      SetWindowWord( hwndChild, WE_SHADE, IDM_100 );
      SetWindowWord( hwndChild, WE_COLOR, wType );
      MdiSetAccel( hwndChild, hAccel );
   }

   return hwndChild;
}

/*
 * ColorWndProc( hwndColor, message, wParam, lParam ) : LONG;
 *
 *    hwndChild      Bezug des MDI-Hauptfensters
 *    message        übergebene Nachricht
 *    wParam         WORD-Parameter der Nachricht
 *    lParam         LONG-Parameter der Nachricht
 *
 * verarbeite Nachrichten an eines der Dokument-Tochterfenster. Die
 * WM_COMMAND-Nachrichten der zugeordneten und angezeigten Menüleiste werden
 * ebenfalls an diese Funktion übergeben.
 */
```

Listing 10: *(Fortsetzung)*

```
long FAR PASCAL ColorWndProc(

   HWND        hwndChild,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   char        szText[20];        /* Text im Abbildungsbereich */
   HBRUSH      hBrush;            /* Farbmuster zum Füllen des Dokuments */
   PAINTSTRUCT Paint;             /* Zeichnen-Datenstruktur */
   int         wColor;            /* Farbe des aktuellen Dokuments */
   int         wShade;            /* Grauton des aktuellen Dokuments */

   switch ( message )
   {
   case WM_COMMAND:
      switch( wParam )
      {
      /* Menü "Datei" */
      case IDM_SAVE:
      case IDM_SAVEAS:
      case IDM_PRINT:
         break;
      case IDM_CLOSE:
         PostMessage( hwndChild, WM_SYSCOMMAND, SC_CLOSE, lParam );
         break;
      case IDM_0:
      case IDM_25:
      case IDM_50:
      case IDM_75:
      case IDM_100:
         CheckMenuItem( MdiGetMenu( hwndChild ),
            SetWindowWord( hwndChild, WE_SHADE, wParam ),
            MF_UNCHECKED );
         CheckMenuItem( MdiGetMenu( hwndChild ),
            GetWindowWord( hwndChild, WE_SHADE ),
            MF_CHECKED );
         InvalidateRect( hwndChild, (LPRECT) NULL, TRUE );
         break;
      }
      break;

   case WM_ERASEBKGND:
      return TRUE;

   case WM_PAINT:
      wColor = GetWindowWord( hwndChild, WE_COLOR );
      wShade = GetWindowWord( hwndChild, WE_SHADE );
      BeginPaint( hwndChild, &Paint );
      hBrush = CreateSolidBrush( rgbColors[wColor][wShade - IDM_0] );
      FillRect( Paint.hdc, &Paint.rcPaint, hBrush );
      DeleteObject( hBrush );
      strcpy( szText, szShadings[wShade - IDM_0] );
      TextOut( Paint.hdc, 0, 0, szText, strlen( szText ) );
      EndPaint( hwndChild, &Paint );
      break;
   }
   return MdiChildDefWindowProc( hwndChild, message, wParam, lParam );
}

/*
 * MainDlgNew( hDlg, message, wParam, lParam ) : int;
 *
 *    hDlg            Bezug auf Dialogfeld-Fenster
 *    message         übergebene Nachricht
 *    wParam          WORD-Parameter der Nachricht
 *    lParam          LONG-Parameter der Nachricht
 *
 * verabeite Nachrichten vom Dialogfeld des Befehls "Neu"
 */

int FAR PASCAL MainDlgNew(
   HWND        hDlg,
```

Listing 10: *(Fortsetzung)*

```
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   static int  iButton;           /* Position der runden Optionsfelder */
   int         iReturn = FALSE;   /* Rückgabewert */

   switch ( message )
   {
   case WM_INITDIALOG:
      SendMessage( hDlg, WM_COMMAND, DLGNEW_RED, 0L );
      iReturn = TRUE;
      break;

   case WM_COMMAND:
      switch( wParam )
      {
      case DLGNEW_RED:
      case DLGNEW_GREEN:
      case DLGNEW_BLUE:
         iButton = wParam;
         CheckRadioButton( hDlg, DLGNEW_RED, DLGNEW_BLUE, iButton );
         if ( HIWORD( lParam ) == BN_DOUBLECLICKED )
         {
            SendMessage( hDlg, WM_COMMAND, IDOK, 0L );
         }
         break;

      case IDOK:
         EndDialog( hDlg, iButton );
         break;

      case IDCANCEL:
         EndDialog( hDlg, 0 );
         break;
      }
      break;

   }
   return iReturn;
}

/*
 * MainDlgAbout( hDlg, message, wParam, lParam ) : int;
 *
 *    hDlg            Bezug auf Dialogfeld-Fenster
 *    message         übergebene Nachricht
 *    wParam          WORD-Parameter der Nachricht
 *    lParam          LONG-Parameter der Nachricht
 *
 * verabeite Nachrichten vom Dialogfeld des Befehls "Über COLORS"
 */

int FAR PASCAL MainDlgAbout(
   HWND        hDlg,
   unsigned    message,
   WORD        wParam,
   LONG        lParam )
{
   int         iReturn = FALSE;  /* Rückgabewert */

   switch( message )
   {
   case WM_INITDIALOG:
      iReturn = TRUE;
      break;

   case WM_COMMAND:
      EndDialog( hDlg, TRUE );
      break;
   }
   return iReturn;
}
```

Listing 10: *(Ende)*

CD-ROM setzt sich nun auch in Deutschland durch:

Das Gigabyte-Zeitalter

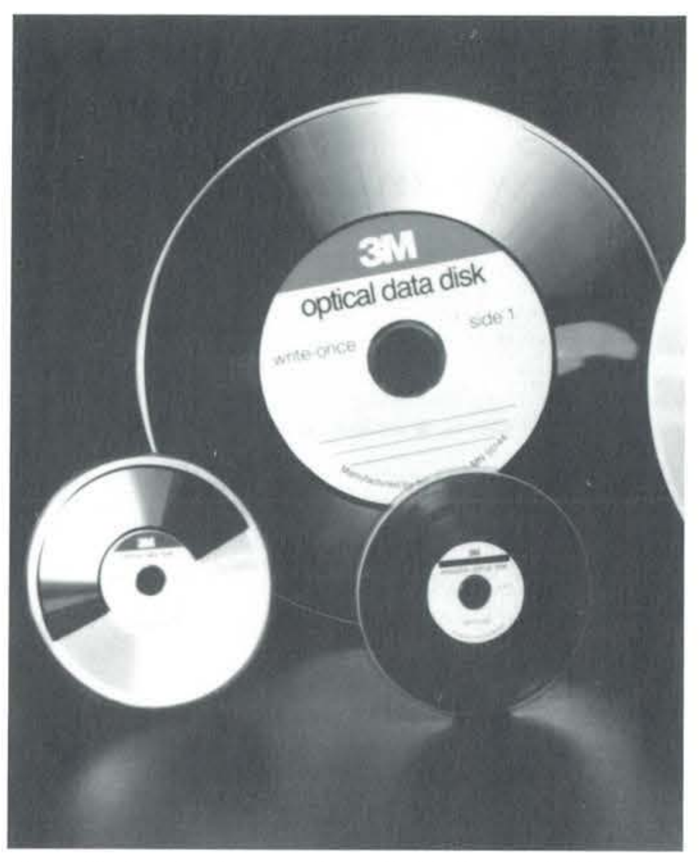
Komplexe Fertigungssysteme, überregionale Reglementierungen und Verwaltungsverfahren machen Informationen in steigendem Maß zu einem Wirtschaftsfaktor. In USA ist man sich des enormen Wettbewerbsvorteils durch komprimierte Datenbanken mehr bewußt als in Europa, weshalb dort die CD-ROM ein stark verbreitetes Speichermedium ist.

Die CD-ROM wird, analog der CD im Audiobereich, vom Hersteller mit Software und Daten »beschrieben«. Sie verfügt über Datenkapazitäten, die da anfangen, wo andere Massenspeicher aufhören, nämlich ungefähr bei 550 Mbyte. Diese Kapazität entspricht annähernd 3 Mio. A4-Seiten Text oder 1000 Disketten. Der Nachteil: CD-ROMs können nur einmal beschrieben und dann nicht mehr verändert oder gelöscht werden. Der CD-ROM-Anwender benötigt einen IBM-kompatiblen PC mit mindestens 512 Kbyte Hauptspeicher einen Controller, um das CD-ROM-Laufwerk anzuschließen.

Die Produktion der CD-ROM läßt sich in zwei wesentliche Abschnitte gliedern: die Informationsaufbereitung und der eigentliche Herstellungsprozeß der CD. Zur Festlegung des Datenbankdesigns setzt sich der Anbieter von Abfragesprachen (Retrievalsoftware) mit dem Kunden zusammen. Im nächsten Schritt wird die zwischen Kunde und Hersteller vereinbarte Softwarelösung für die Datenbank geschaffen. Das künftige Datenbankdesign wird dann, versehen mit Testdaten, auf Magnetspeichern simuliert. Erst nachdem sich die Datenbank auf diese Weise bewährt hat, wird sie zusammen mit den erforderlichen Daten auf CD gepreßt.

Die Speicherung erfolgt bei optischen Laufwerken mittels Laser eingebrannter Informationen, sogenannter Pits. Gelesen werden diese Informationen berührungslos durch einen fokussierten Laserstrahl. Die hohe Aufzeichnungsdichte hat einerseits den Vorteil, daß große Datenmengen auf einer Compact Disk untergebracht werden können. Andererseits ist die Datensicherheit um ein vielfaches höher als bei magnetischen Speichern, wo Head-Crashes den gesamten Datenbestand einer Festplatte vernichten können. Da die informationstragende Schicht einer CD in Kunststoff eingebettet liegt, ist sie bei weitem nicht so empfindlich gegen Schmutzpartikel wie magnetische Datenträger.

Entgegen aller optimistischen Prognosen hat sich der CD-ROM-Markt in Deutschland bis heute noch nicht in dem Maß entwickelt, wie beispielsweise in USA oder in Italien. Der Hauptgrund ist sicher, daß hierzulande kaum attraktive CD-Titel im Handel sind. Mehr als die Hälfte der deutschen CD-Titel sind Nachschlagewerke, die auch in gedruckten Versionen vorliegen. Geht es um die verkauften Stückzahlen, so sind die Werte für die Anbieter enttäuschend: Haben Buchversionen, beispielsweise von Adreßbuchverlegern, Auflagen zwischen 20.000 und 30.000 Stück,



Die optischen Speichermedien CD-ROM und WORM im Größenvergleich.

erreichen die CD-Publikationen oftmals nicht einmal eine Auflage von 1.000. Da es noch keinen richtigen Markt für CD-ROM-Anwendungen gibt, haben sich auch die Preise noch nicht stabilisiert. Ein Teufelskreis: teilweise recht teure Anwendungen verhindern den Absatz, den die Entwickler brauchen, um mehr und somit billigere Anwendungen auf den Markt bringen zu können.

Deutsche CD-Software-Entwickler konzentrieren sich nach den Erfahrungen der letzten Jahre auf »geschlossene Benutzergruppen, zum Beispiel Informationsanbieter in der Industrie, die ihrer Vertriebs- und Service-Organisation umfangreiche Informationen zur Verfügung stellen müssen«, so Manfred P.Wendt von der eps GmbH (Tochterunternehmen der Bertelsmann-Gruppe). Der Trend wird also dahin gehen, daß im Unternehmen bereits vorhandene Daten aufbereitet und auf CD-ROM in größerem Rahmen im Unternehmen genutzt werden. Verstärkt wird diese Tendenz dadurch, daß inzwischen sehr gute Retrieval-Software angeboten wird und Spezialanwendungen für Unternehmen deshalb nicht mehr so teuer sind.

Neben den unternehmensspezifischen Anwendungen werden in Deutschland hauptsächlich die CD-Produkte großer Verlage genutzt, die hierzulande die Hauptanbieter

von CD-ROMs sind. Als 1986 in USA erstmalig »Grolier's Encyclopedia« auf CD-ROM erschien, wurde begeistert von einem neuen Zeitalter der Lexikographie gesprochen und Grolier machte in kürzester Zeit einen unerwartet hohen Umsatz. In Deutschland werden hauptsächlich Datensammlungen aus dem Bereich der Wirtschaft, wie »Wer liefert was?«, »Handbuch der Großunternehmen« von Hoppenstedt oder die internationale Bibliographie »International Books In Print« des Saur-Verlags angeboten. Es gibt aber auch eine juristische Bibliothek (»Juris«, C.H.Beck/entwickelt von Dataware 2000), die alle Entscheidungen und Aufsätze der »Neuen Juristischen Wochenzeitschrift« enthält, die in der Leitsatzkartei ab Heft 1/85 abgedruckt sind. Bei dieser CD-Anwendung spielt, ebenso wie beispielsweise bei dem »Gefahrengut«-Katalog auf CD, die Verknüpfungsmöglichkeit einzelner Felder die Hauptrolle. Das bedeutet, der Anwender kann innerhalb eines Arbeitsvorgangs auf mehrere Begriffe zugreifen und Daten in kurzer Zeit nach individuellen Gesichtspunkten selektieren.

Diese Selektionsmöglichkeit will man jetzt auch in Multimedia-Anwendungen auf CD-ROM realisieren. Fachleute sehen in der Kombination von Texten, Tönen, Stand- und Bewegtbildern eine Chance, den CD-ROM-Markt zu einem enormen Wachstumsmarkt zu etablieren. In USA haben derartige Anwendungen bereits heute eine große Verbreitung im Schulungssektor. Entwicklungsansätze in diesem Bereich gibt es schon länger, zum Beispiel das ursprünglich am David-Sarnoff-Forschungszentrum in Princeton entwickelte »Digital Video Interactive« (DVI) oder das bekanntere »CD-Interactive« (CD-I), eine gemeinsame Entwicklung von Philips und Sony. Da man CD-I-Applikationen künftig auch mit CD-ROM-Laufwerken nutzen will, haben sich im letzten Jahr Philips, Sony und Microsoft auf die erweiterte CD-ROM Architektur »CD-ROM XA« (Extended Architecture) geeinigt.

Der offizielle Standard für das (logische) Aufzeichnungsformat, die Directory-Struktur und die Schnittstellen auf CD-ROM wurde bereits zu Beginn der CD-ROM-Entwicklung von einer Gruppe von Softwareherstellern und Hardwareproduzenten festgelegt. Diese Gruppe nennt sich nach ihrem Tagungsort »High Sierra Group«. Was die Controller-Schnittstelle angeht, verwenden fast alle Hersteller von CD-ROM-Laufwerken die SCSI-Schnittstelle, die sich allmählich auch im Bereich der Magnetspeicher durchsetzt. Neben der hohen Datentransferrate hat SCSI den Vorteil, daß man bis zu sieben zusätzliche Laufwerke an einen Controller anschließen kann.

Trotz all dieser Standardisierungsbemühungen gibt es aber noch einige Inkompatibilitäten zu beklagen. Die eben erwähnte SCSI-Schnittstelle beispielsweise ist nicht bei allen Laufwerken identisch, so daß Übertragungsprobleme auftauchen können. Es kommt vor, daß sich die Treiber für einzelne CDs gegenseitig stören. Die Cartridges sind nicht genormt, so daß es hier Probleme beim Einlegen in das Laufwerk geben kann.

Im Laufwerksbereich gibt es nicht viele Anbieter in Deutschland. Den allergrößten Teil des Marktes beherrschen Hitachi, Sony, Toshiba und Phillips. Die CD-ROM-Laufwerke gibt es in externen Versionen oder als 5,25-Zoll-Einbaugeräte in voller Bauhöhe. Die Preise liegen zwischen 1500 Mark und 7000 Mark, wobei keine Korrelation zwischen dem Wert des Produktes und dessen Position innerhalb des Preisbereichs zu bestehen scheint.

Die Preise für die CDs orientieren sich selbstverständlich an der Komplexität der Anwendung. Da aber die europäische Audio-CD-Industrie in der letzten Zeit teilweise unter Überkapazitäten leidet, ist anzunehmen, daß der aufkommende Wettbewerb wahrscheinlich geringere Aufbereitungs- und Pressungskosten für CD-ROMs zur Folge haben wird.

Frischer Wind weht aber auch aus einer anderen Ecke: Einige Hersteller, wie Apple und Atari haben sich entschlossen, ihre PCs auf Wunsch mit CD-ROM-Laufwerk auszuliefern. Gerüchte aus Fachkreisen besagen, daß bis zum Herbst noch einige andere PC-Hersteller ihren Kunden diese Möglichkeit bieten. Apple-Gründer Steven Jobs hat bereits im vergangenen Jahr auf der Herbst-Comdex in Las Vegas seine Eigenentwicklung, den NeXT-Computer, standardmäßig mit einem magneto-optischen Speicher ausgestattet.

Dominique Guignol

Was gibt es auf CD-ROM?

Bertelsmann

- Die Bibel auf deutsch, englisch, etc.
- ABDA-Datenbank für Apotheken: 150.000 Arzneimittel, Hersteller, Preis
- Liefern und Leisten - Das deutsche Branchen-Fernsprechbuch
- Silverdat - Das Datensystem für die Automobilwirtschaft
- Gemeinschafts-CD-ROM/Orts- und Straßenverzeichnis der Deutschen Bundespost sowie Müllers Großes Deutsches Ortsbuch

Dataware 2000

- Hoppenstedt Firmenreport - 430.000 Firmenprofile

Microsoft Bookshelf

- World Almanac of Books and Facts
- Rogets II Electronic Thesaurus
- The American Heritage Dictionary
- The Chicago Manual of Style
- Bartletts Familiar Quotations

NeXT

- Websters Collegiate Dictionary
- Websters Collegiate Thesaurus
- Shakespeares gesammelte Werke

CDs von Microsoft:

Bookshelf und Programmer's Library

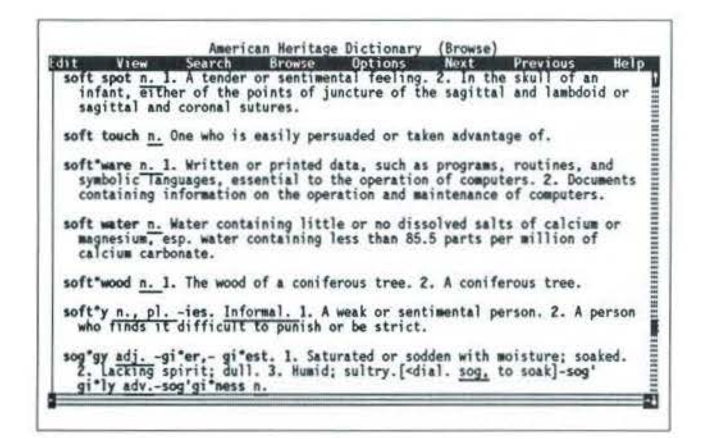


Bild 1: *Hier wurde im Wörterbuch von Bookshelf nach dem Wort »Software« gesucht.*

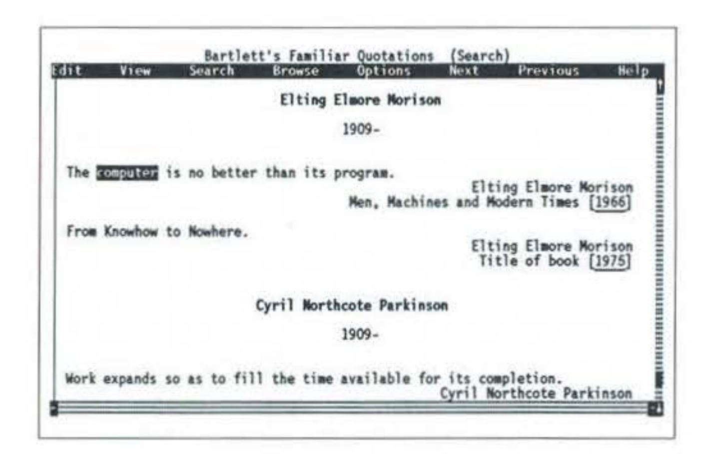


Bild 2: *In der Zitatensammlung von Bookshelf findet man auch interessante Aussagen über Computer.*

Microsoft ist schon seit den Anfängen der CD-ROM-Technologie ein Vorreiter auf diesem Gebiet. Die Nachschlagewerk-CD Bookshelf war die erste CD-Anwendung, die auf breiter Basis einen Erfolg verbuchen konnte. Mit der Programmer's Library bietet Microsoft seit vergangenem Jahr auch eine ausgezeichnete Daten- und Dokumentationssammlung für Programmierer an.

Microsoft Bookshelf

Microsoft Bookshelf versammelt auf einer einzigen CD eine ganze Reihe von wichtigen Handbüchern und Nachschlagewerken, die gezielt für den Bedarf von Journalisten, Schriftstellern und Publizisten ausgewählt worden sind.

- *The American Heritage Dictionary*, ein elektronisches Wörterbuch mit über 200.000 Einträgen (*Bild 1*).
- *Roget's II: Electronic Thesaurus*, ein Synonymwörterbuch mit über 500.000 Synonymen.
- *The 1987 World Almanac and Book of Facts*, Informationen aus den Bereichen Politik, Medien und Sport.
- *Bartlett's Familiar Qutations*, eine Zitatensammlung mit über 22.500 klassischen und aktuellen Zitaten (*Bild 2*).
- *The Chicago Manual of Style*, Schreibanleitung und Stilführer für Schreibende.
- *Houghton Mifflin Spelling Verifier and Corrector*, Rechtschreibprüfer.
- *Forms and Letters*, eine umfangreiche Sammlung nützlicher Formulare, Briefe und Gliederungen.
- *U.S. ZIP Code Directory*, das Postleitzahlenverzeichnis für die USA.
- *Houghton Mifflin Usage Alert*, Verwendunghinweise für Wörter.
- *Business Information Sources*, eine umfassende Sammlung von Informationen aus den Bereichen Geschäft, Wirtschaft, Management und Marketing.

Bookshelf kann mit den meisten Anwendungsprogrammen verwendet werden, so ist es von vorneherein darauf ausgerichtet, mit 12 weitverbreiteten Textverarbeitungen (z.B Microsoft Word, PC-Write, DisplayWrite, WordStar usw.) und Programmen wie Microsoft Multiplan oder Lotus 1-2-3 Daten auszutauschen. Man braucht nur den Cursor auf ein Wort zu positionieren, zu dem man nähere Informationen benötigt, und auf Tastendruck werden diese Informationen angezeigt. Die Informationen können dann in den Text kopiert werden.

Für Programmierer dürfte Bookshelf weniger interessant sein, für diese Zielgruppe bietet Microsoft seine Programmer's Library an.

Programmer's Library

Mit der Microsoft Programmer's Library steht Entwicklern von PC-Software ein leistungsfähiges Nachschlagewerk zur Verfügung, das auf einer CD-ROM mehr als 20.000 Seiten Dokumentation und über 1.000 Beispielprogramme bereitstellt. Die 48 Handbücher sowie andere technische Dokumentationen der Programmer's Library - gespeichert auf einer CD - beinhalten Informationen über die Betriebssysteme und Programmiersprachen von Microsoft. Das Spektrum reicht dabei von relativ kurzen Hinweisen bis zu detaillierten Abhandlungen. Für die gesamte Text-Datenbank gibt es ein Inhaltsverzeichnis und Querverweise, so daß das Auffinden und Verbinden von Informationen relativ einfach ist. Die Texte lassen sich mit Hilfe eines Texteditors oder eines Textverarbeitungsprogramms vom Anwender aufrufen und direkt in Programme oder Dokumente hineinkopieren. Direkt werden die gängigsten Programm-Editoren sowie fünf Textverarbeitungsprogramme unterstützt. Auch die Übernahme in Form von ASCII-Files ist möglich.

Die CD-ROM-Disk enthält darüber hinaus eine Reihe von Informationen über Hardware, wie die Microsoft-Maus, CD-ROM-Laufwerke und Videokarten.

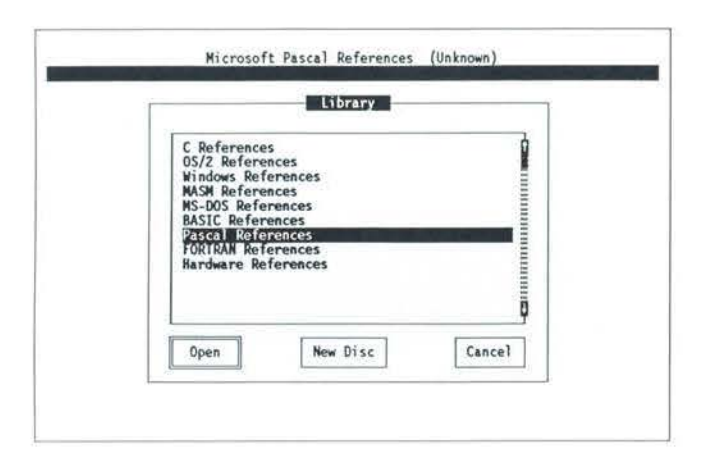


Bild 3: *Das Auswahlmenü, aus dem heraus der Bereich gewählt werden kann, in dem man suchen will.*

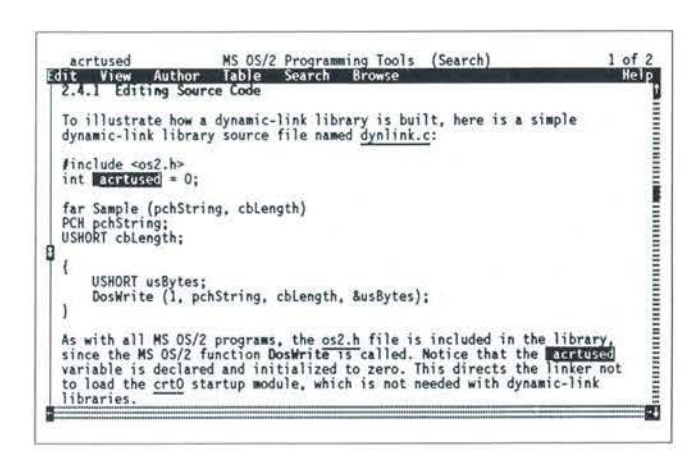


Bild 4: *Gesuchte Wörter werden auf dem Bildschirm (hier _acrtused) hervorgehoben.*

Die Programmer's Library läßt sich sowohl direkt als auch aus anderen Programmen aufrufen. Im Gegensatz zu speicherresidenten Programmen kann der Anwender das Programmer's Library-Programm jedoch auf einfache Weise wieder aus dem Speicher löschen, um Platz für andere Programme oder Daten zu schaffen.

Nutzen aus der Programmer's Library können nicht nur Programmierer ziehen, sondern alle, die mit Software arbeiten, z.B. Autoren technischer Beiträge oder Produkt-Support-Spezialisten, die schnell auf Informationen zugreifen wollen, ohne umfangreiche Handbücher durchblättern zu müssen. Im einzelnen sind in der Programmer's Library folgende Bücher und Handbücher enthalten:

- *The MS-DOS Encyclopedia*
- Die Handbücher aus dem Software Development Kit (SDK) für OS/2
- Die Handbücher des Microsoft C-Compilers in der Version 5.1
- Das Buch *Proficient C* von Augie Hansen
- Die Handbücher des Microsoft Windows Development Kits in der Version 2.0
- Das Buch *Programming Windows* von Charles Petzold
- Die Handbücher des Microsoft Makroassemblers 5.1
- Das Buch *Advanced MS-DOS* von Ray Duncan
- Das Buch *Programmer's Guide to PC and PS/2 Video Systems* von Richard Wilton
- Die Handbücher von Microsoft Basic, Pascal und Fortran
- Über 1000 Beispielprogramme für MS-DOS, MS OS/2 und Microsoft Windows.

Das Inhaltsverzeichnis der Programmer's Library ist in neun Kapitel gegliedert (*Bild 3*). Jeder Bereich umfaßt alle technischen Handbücher der jeweiligen Version des Betriebssystems oder der Programmiersprache. Die Dokumentation für Presentation Manager und LAN-Manager war für die erste Ausgabe noch nicht verfügbar, ist aber in der neuesten Version 1.1 enthalten (s. »Mitteilungen«).

Kernpunkt der Microsoft Programmer's Library ist eine hochentwickelte Suchfunktion (*Bild 4*). Der Anwender gibt nur verschiedene Suchbegriffe ein und spezifiziert die logischen Beziehungen zwischen ihnen. Dabei läßt sich vorgeben, ob eine Ausgabe erfolgen soll, wenn einige der Begriffe gefunden wurden, oder ob nur dann eine Ausgabe erfolgen soll, wenn alle Begriffe gefunden wurden, oder daß nur dann eine Ausgabe erfolgt, wenn die Suchfunktion auf eine exakt definierte Zeichenfolge gestoßen ist. Man kann aber die Suchfunktion auch so einsetzen, daß verschiedene Begriffe im selben Kapitel oder Absatz zu finden sind.

Wenn die Programmer's Library aus einem Texteditor oder einem Textverarbeitungsprogramm heraus angesprochen wird, so erfolgt die Suche automatisch nach dem Begriff, auf dem der Cursor gerade steht. Der Anwender braucht dazu lediglich die Eingabetaste zu drücken, um die Suchfunktion einzuschalten. Er hat wie bei einem Buch die Möglichkeit, im Inhaltsverzeichnis ebenso wie in den aufgerufenen Texten zu »blättern«, um sich einen Überblick zu verschaffen. Damit der Suchaufwand und die Suchzeit in vernünftigen Grenzen bleiben, erfolgt die Suche immer in einem Teilbereich des gesamten Inhalts. Der Anwender hat jedoch die Möglichkeit, schnell zwischen den Teil-Inhaltsverzeichnissen und den Bereichen hin- und herzuschalten, indem er die dafür zur Verfügung stehenden Such- und Bibliothekskommandos benutzt.

Die Kompatibilität der Programmer's Library zu einer Vielzahl von Texteditoren und Textverarbeitungsprogrammen macht es möglich, daß Informationen in das Anwendungsprogramm übernommen werden.

Ganz gleich, wieviele oder welche Programme Sie schreiben, die Programmer's Library dürfte innerhalb kürzester Zeit zum wertvollsten Nachschlagewerk bei der Programmierung werden. Hier findet man so gut wie alles, was man zur Programmierung auf PCs benötigt. Darüber hinaus hat man den Vorteil daß man mit CDs benötigte Informationen wesentlich einfacher und gleichzeitig schneller finden kann, als in Büchern.

OS/2-Textbildschirme auf Knopfdruck speichern:

Vom Regen in die Sonne: Monitore unter OS/2

In den letzten Ausgaben des Microsoft System Journals haben sich mehrere Artikel mit der Programmierung von speicherresidenten Programmen unter MS-DOS und den dabei zu bewältigenden Schwierigkeiten beschäftigt. Wir wollen mit diesem Beitrag einen Blick in die Betriebssystemzukunft werfen und Ihnen die Möglichkeit vorstellen, unter OS/2 TSR-ähnliche Programme zu realisieren.

Aus dem Leben eines MS-DOS-Anwenders sind sie nicht mehr wegzudenken: die zahlreichen speicherresidenten Programme, die auf Knopfdruck nützliche Utilities aktivieren und von trickreichen Programmierern unter Verwendung von teilweise nicht dokumentierten Betriebssystemfunktionen erstellt wurden. Die TSRs (TSR = terminate and stay resident) unter MS-DOS versuchen, dem Betriebssystem, das nicht als Multitasking-Umgebung entwickelt und konzipiert wurde, scheinbar Multitasking-Fähigkeiten abzugewinnen; scheinbar deshalb, weil während der Aktivitäten des TSR-Programms die Anwendung, aus der heraus das speicherresidente Programm aktiviert wurde, unterbrochen wird. Ein zentrales Element dieser Programme bildet eine Routine, mit der die Tastatureingaben abgefangen werden, um zu überprüfen, ob mit der eingegebenen Taste die Pop-up-Utility aktiviert werden soll. Darüber hinaus muß besonders sorgfältig die Verwendung von Interrupt-Aufrufen implementiert werden (beispielsweise um eine Datei anzulegen), da MS-DOS nicht reentrant ist, und somit die Betriebssystemfunktionen nicht von mehreren Programmen zeitgleich benutzt werden können.

Das Zusammenspiel von mehreren, gleichzeitig geladenen TSR-Programmen endet oft in einer Tragödie, da manche Pop-ups nicht mit bestimmten Anwenderprogrammen geladen werden können oder speicherresidente Programme untereinander inkompatibel sind, so daß sie nicht gemeinsam benutzt werden können.

OS/2 hingegen wurde als Multitasking-Betriebssystem konzipiert und besitzt damit die Fähigkeit, mehrere Prozesse, die gleichzeitig ablaufen, zu steuern, zu unterstützen und zu kontrollieren. Dies macht die Programmierung von TSR-ähnlichen Applikationen erheblich einfacher. Der Software-Entwickler wird bei diesen Projekten außerdem durch gut dokumentierte Betriebssystemfunktionen und Schnittstellen zum OS/2-API (application program interface, Programmierschnittstelle) unterstützt, mit denen auch ohne die von Kaare Christian beschriebene Macho-Philosophie (vgl. Speicherresidente Programme in MS-C, in MSJ Nov./Dez. 1988, S. 14 ff.) gut funktionierende und kooperative Pop-up-Utilities erstellt werden können.

Den Gerätemonitoren, einem Mechanismus, mit dem unter OS/2 Programme erstellt werden können, die den speicherresidenten Programmen unter MS-DOS ähneln, ist dieser Artikel gewidmet.

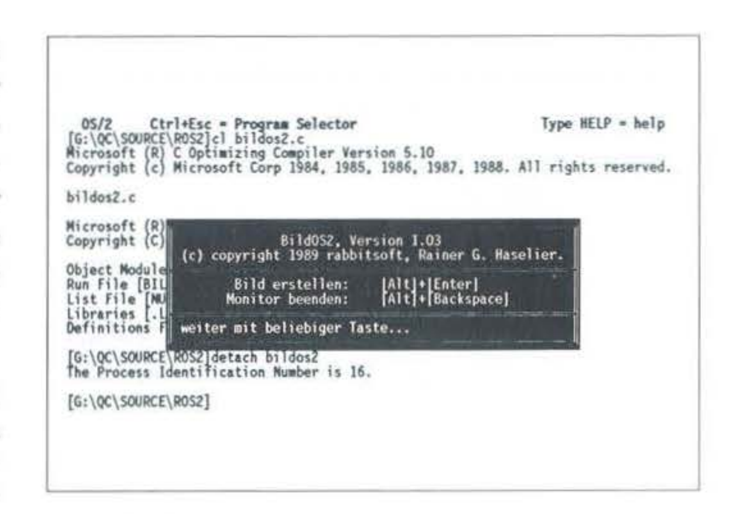


Bild 1: *Die Installationsmeldung von BILDOS2.*

Wir stellen Ihnen die konzeptionellen Grundlagen, die wichtigsten API-Funktionen zur Implementierung von Gerätemonitoren und ein nützliches Programm vor, mit dem Sie unter OS/2 Hardcopies von Textbildschirmen erstellen können, die dann mit dem in der März/April-Ausgabe des MSJ vorgestellten Programm BW.EXE in das WORD-Dateiformat konvertiert werden können.

Bedienung von BILDOS2

BILDOS2 wird mit dem OS/2-Befehl DETACH als Hintergrundprozeß gestartet. Rufen Sie von der OS/2-Befehlsebene das Programm folgendermaßen auf:

```
[C:\]detach bildos2
```

Das Programm wird nun als Hintergrundprozeß für die aktive ScreenGroup installiert, ein Info-Fenster informiert Sie über diesen Vorgang (*Bild 1*). Ab diesem Zeitpunkt können Sie durch Eingabe der Tastenkombination [Alt] [Enter] den Inhalt des aktuellen Textbildschirms in eine Datei schreiben lassen. Hierbei werden auch die Attribute der Zeichen übernommen, damit Sie die so erhaltenen Bild-Dateien mit dem Programm BW.EXE (vgl. MSJ März/April 89) in das Word-Dateiformat konvertieren können. (Wenn Sie die Konvertierung der Bild-Dateien unter OS/2 durchführen wollen, ist es erforderlich, das Programm BW.EXE mit der OS/2-Version des MSC 5.1-Compilers neu aufzubauen. Am Quellcode von BW.C brauchen dazu jedoch keine Änderungen vorgenommen werden.)

Die Bild-Dateien werden von BILDOS2 in dem Verzeichnis abgelegt, aus dem heraus Sie das Programm gestartet haben. Auch ein späterer Wechsel des Verzeichnisses hat darauf keinen Einfluß. Der Versuch, BILDOS2 in der gleichen ScreenGroup ein zweites Mal zu installieren, endet mit einer Fehlermeldung. Sie können BILDOS2 jederzeit durch Eingabe der Tastenkombination [Alt] [Backspace] wieder entfernen.

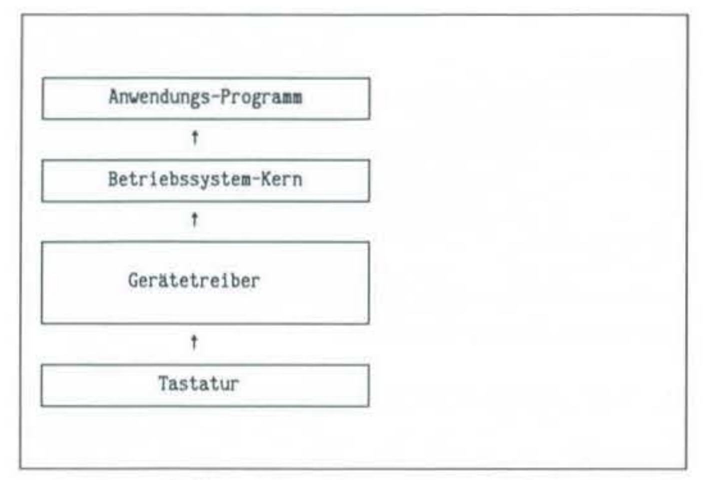


Bild 2: *Datenfluß ohne installierten Tastatur-Monitor.*

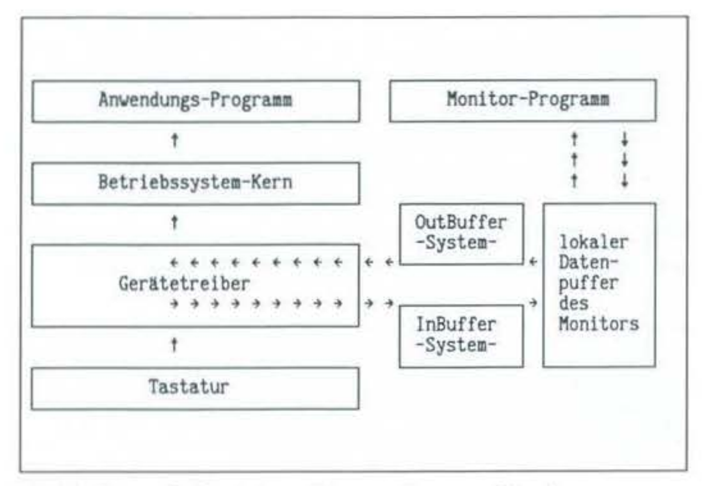


Bild 3: *Datenfluß mit installiertem Tastatur-Monitor.*

Der Datenstrom unter OS/2

Was ist nun ein Gerätemonitor? Ein Gerätemonitor ist ein Programm, das bestimmte Funktionen des OS/2 API benutzt, um Zugriff auf den Datenstrom eines Gerätetreibers zu erhalten und ihn überwachen zu können (monitoring). Monitore können für folgende OS/2-Gerätetreiber installiert werden: Tastatur, Maus und Drucker. Bei den blockorientierten Geräten (Platte) ist die Verwendung eines Monitors nicht möglich. Der Übersichtlichkeit wegen beschränken wir uns bei den folgenden Ausführungen auf Gerätemonitore, die die Tastatur überwachen.

Die Möglichkeit, daß beispielsweise der Datenstrom von der Tastatur problemlos abgefangen werden kann, hängt mit der Organisation des Datenflusses unter OS/2 zusammen, den wir uns nun ansehen wollen (*Bild 2*).

Das von der Tastatur eingelesene Zeichen wird zuerst an den Gerätetreiber weitergereicht. Von dort aus erreicht es den Betriebssystemkern, der beispielsweise mit der Funktion `KbdCharIn()` von einer Anwendung angewiesen wurde, ein Zeichen einzulesen. Auf diesem Wege wird das Zeichen weiterbearbeitet und gefiltert. Die Tastenkombinationen, mit denen beispielsweise der Session-Manager aktiviert oder zu einer anderen Screen-Group umgeschaltet werden kann, werden auf diesem Wege verschluckt und bleiben dem Anwendungsprogramm verborgen. Das Geheimnis der Funktionsweise eines Monitors besteht darin, das die Daten nie vom Gerät direkt an ein Anwendungsprogramm gehen, sondern immer die Umleitung über das Betriebssystem fahren müssen.

Ist ein Monitorprogramm installiert, sieht der Datenfluß ein wenig anders aus (*Bild 3*). Wenn ein Zeichen über die Tastatur eingegeben wird, ist der Gerätetreiber der erste, der über das eintreffende Zeichen informiert wird. Da nun ein Monitorprogramm installiert ist, das die eintreffenden Tasten erhalten soll, werden die Tastaturdaten vom Betriebssystem zuerst in einen vom Monitorprogramm bereitgestellten Eingabepuffer kopiert.

Das Monitorprogramm kann die eingehenden Daten lesen, wodurch sie aus dem Eingabepuffer in den lokalen Datenpuffer des Monitors kopiert werden. Dort - und auch nur dort - darf das Monitorprogramm die Daten untersuchen. Entdeckt das Programm einen der Hot-Keys, kann es die zugehörige Programmfunktion aktivieren. Dazu gehört auch die Möglichkeit, die Taste verändert weiterzureichen (beispielsweise Klein- in Großbuchstaben umzuwandeln oder umgekehrt) oder sie überhaupt nicht mehr an den Gerätetreiber zurückzuschicken.

Wenn die untersuchte Taste unverändert oder modifiziert weitergereicht werden soll, wird sie, durch Aufruf einer entsprechenden API-Funktion, aus dem lokalen Datenpuffer des Monitors in den Ausgabepuffer, der vom Betriebssystem kontrolliert wird, kopiert. Sind für das gleiche Gerät weitere Monitore installiert, werden die Daten, bevor sie das sich im Vordergrund befindende Programm erreichen, von einem zum nächsten Monitor weitergereicht.

Um den beschriebenen Datenfluß zu ermöglichen, muß das Monitorprogramm drei Datenpuffer einrichten, die alle die ein- bzw. ausgehenden Daten erhalten. Das Format dieser Puffer, die von Gerätemonitoren benutzt werden, ist genau definiert. Am Beispiel der Tastatur-Monitore wollen wir sie uns genauer ansehen.

Die Datenstrukturen für Tastatur-Monitore

Das Format der Ein- und Ausgabepuffer, die vom Betriebssystem benutzt werden, ist ganz einfach. In den ersten 16 Bit muß die Größe des Puffers enthalten sein. Der Puffer selbst sollte mindestens 64 Byte groß sein, wobei jedoch 128 Byte empfehlenswert sind. Wir gehen hier auf Nummer Sicher und geben die Größe des Buffers in der Strukturdeklaration von `MONBUFF` mit 128 Byte an (*Bild 4*).

Diese - und auch die nächste Struktur `KEYPACKET` - ist in den Header-Dateien des OS/2-SDK nicht enthalten. Wenn Sie häufiger Monitor-Applikationen erstellen, ist es also ratsam, sie in eine eigene Header-Datei aufzunehmen.

```
typedef struct tag_MonBuff {  /* Struktur Ein-/Ausgabepuffer */
   unsigned  length;              /* Länge von MONBUFF      */
   unsigned char buffer[128];     /* vom System benutzt     */
} MONBUFF;
```

Bild 4: *Die Struktur* `MONBUFF` *dient der Datenübertragung zwischen dem Gerätetreiber und dem Monitorprogramm.*

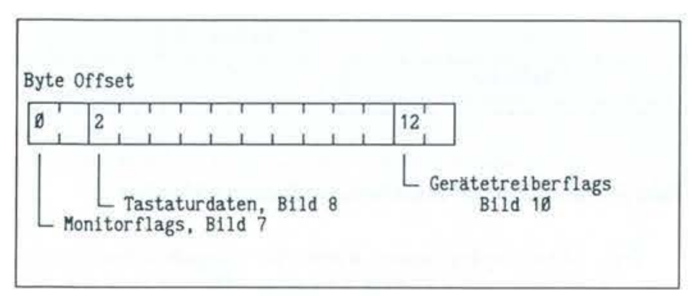


Bild 5: *Das Datenpaket für Tastatur-Monitore.*

```
typedef struct tag_KeyPacket{ /* Struktur Tastatur-Datenpack */
   unsigned         monflags;        /* Monitor Flag-Word      */
   unsigned char    ascii;           /* Taste: ASCII-Code      */
   unsigned char    scan;            /* Taste: Scan-Code       */
   unsigned         nls;             /* Sprachensupport        */
   unsigned         shift;           /* Status Shift-Tasten    */
   unsigned long    keytime;         /* Zeit Tastendruck       */
   unsigned         ddflags;         /* Flags Gerätetreiber    */
} KEYPACKET;
```

Bild 6: *Der Struktur* `KEYPACKET` *können Informationen über die eingegebene Taste entnommen werden.*

Die beiden Variablen `MonInBuf` und `MonOutBuf` vom Typ `MONBUFF` werden im Listing (Abschnitt »globale Daten«) definiert und im Initialisierungsteil der Funktion `main()` wird in das erste Word der Struktur die Größe des Puffers eingetragen. Der Aufbau des lokalen Datenpuffers wird durch die Informationen bestimmt, die bei jedem Tastendruck an den Monitor übermittelt werden (*Bild 5*).

Jedes Tastatur-Datenpaket ist 14 Byte groß, wobei sich in den ersten beiden Bytes das Monitor-Flag-Word befindet, das in ein wenig veränderter Form bei allen Monitor-Datenpaketen Verwendung findet. Ab Offset 2 befinden sich die eigentlichen Tastaturdaten. Dieses Feld ist 10 Byte groß und entspricht der Struktur `KBDKEYINFO`, die in der Header-Datei `bsesub.h` definiert ist. Die gleiche Struktur wird beispielsweise von der API-Tastatur-Eingabefunktion `KbdCharIn()` verwendet. Last but not least finden wir ab Offset 12 das Gerätetreiber-Flag-Word, dem wir zusätzliche Informationen entnehmen können. Alle Tastatur-Informationen werden über die Struktur `KEYPACKET`, die das in *Bild 6* beschriebene Muster besitzt, in BILDOS2 ausgewertet. Fangen wir mit dem Monitor-Flag-Word an (*Bild 7*).

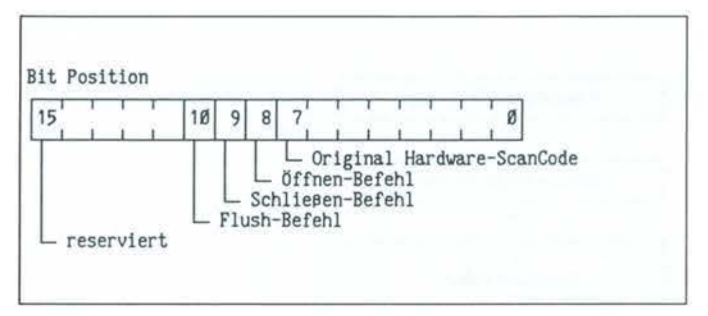


Bild 7: *Das Monitor-Flag-Word.*

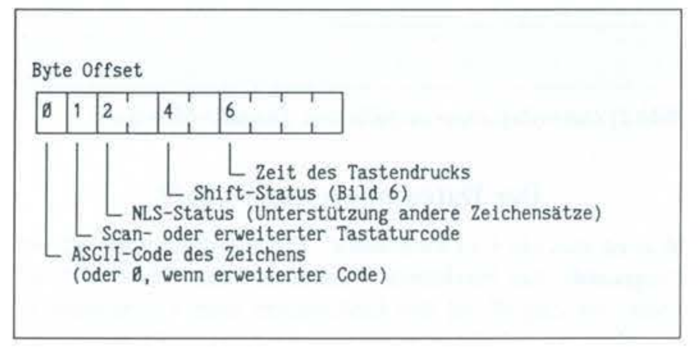


Bild 8: *Die Tastaturdaten.*

Im Monitor-Flag-Word finden wir an den Bit-Positionen 7 bis 0 den Hardware-Scancode, wie er von der Tastatur erzeugt wurde. Bit 8 und 9 sind gesetzt, wenn es sich bei den einkommenden Daten um einen Öffnen- oder Schließen-Befehl handelt. Bit 10 ist von Bedeutung, wenn die Daten innerhalb des Monitors gepuffert werden, beispielsweise bei einem Drucker-Spooler. Ist dieses Bit gesetzt, muß das Monitorprogramm seine Puffer sofort leeren und die Daten weiterreichen. Die Bits 15 bis 11 sind reserviert und sollten nicht benutzt werden.

Diesem Flag-Word entspricht das Strukturelement `monflags` der Struktur `KEYPACKET` (siehe *Bild 6*).

Im ersten Byte der Tastaturdaten (Bild 8, Strukturelement `ascii` von `KEYPACKET`) finden wir den ASCII-Code des eingegebenen Zeichens. Wurde eine Taste mit einem erweiterten Tastaturcode (beispielsweise eine Funktions- oder Cursortaste) eingegeben, befindet sich hier eine 0, und an Byte 1 (Strukturelement `scan`) der Scancode der Taste.

Die Bytes an Offset 2 und 3 (NLS-Status, National language Support) dienen der Unterstützung von Zeichensätzen, die mit dem ASCII-Code nicht erfaßt werden können, sondern zur Darstellung des Zeichensatzes zwei Byte benötigen (DBCS = double byte character set) und bleiben hier unberücksichtigt.

Das Wort ab Offset 4 enthält den Status der Shift-Tasten, der bitweise kodiert und nach dem Muster, das Sie in *Bild 9* sehen, abgelegt ist. (R ist die Abkürzung für Rechts, L für Links.)

Ab Byte 6 wird der Zeitpunkt der Tasteneingabe in Millisekunden nach dem Systemstart angegeben.

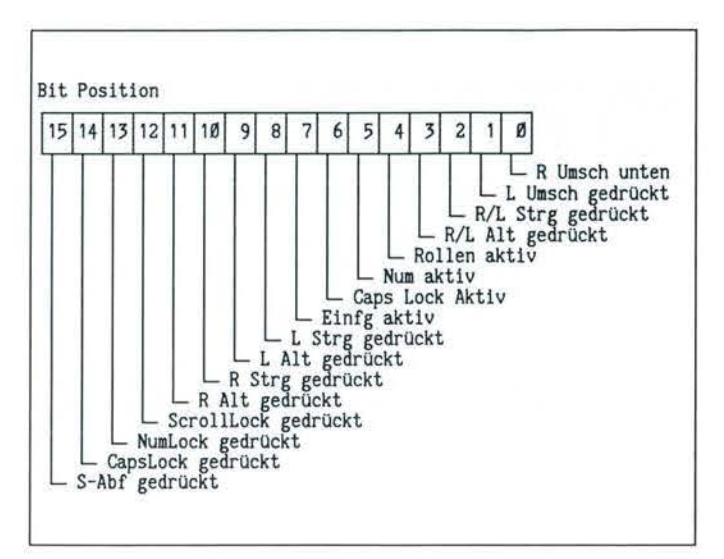


Bild 9: *Das Shift-Status-Word.*

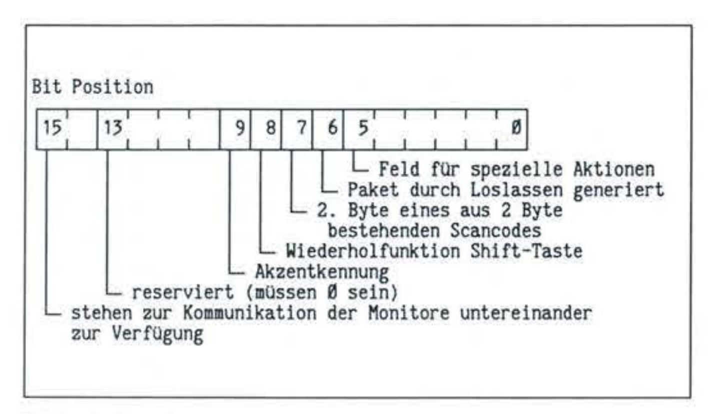


Bild 10: *Die Gerätetreiber-Flags.*

Beim Gerätetreiber-Flag-Word (*Bild 10*) interessiert uns für unsere Zwecke besonders Bit 6, das gesetzt ist, wenn das Datenpaket durch Loslassen einer Taste zustandegekommen ist. Die anderen Bitfelder sind der Vollständigkeit halber nur kurz erwähnt.

Soweit die benötigten Datenstrukturen. Sie sehen, die Informationen, die OS/2 Ihnen zur Verfügung stellt, sind wegen ihrer Vielfalt ein wenig verwirrend. Bei den meisten Tastatur-Gerätemonitoren reicht die Verwendung von drei Daten meist vollkommen aus: der ASCII-Code und der Scan-Code der Taste sowie die Information, ob das Datenpaket durch Loslassen der Taste erzeugt wurde. Auch BILDOS2 kommt mit diesen drei Informationen aus.

Strukturen allein reichen jedoch nicht aus. Wir wollen ja, wie oben beschrieben, den Datenfluß an unser Monitorprogramm umleiten. Und da greift uns das OS/2-API mit den speziell für Gerätemonitore zur Verfügung stehenden Funktionen voll unter die Arme.

```
DosMonOpen(GeräteName, MonHandle);
  char far *GeräteName; /* Name des Gerätes, KBD$, MOU$, PRN */
  unsigned *MonHandle;  /* Adresse, über die von OS/2 eine   */
                        /* Handle zurückgegeben wird         */
```

Bild 11: *Die Funktion* `DosMonOpen( )`.

```
DosMonReg(MonHandle, InBuf, OutBuf, Pos, Index);
  unsigned MonHandle;   /* Handle von DosMonOpen()           */
  char far *InBuf;      /* Eingabe-Puffer für Betriebssystem */
  char far *OutBuf;     /* Ausgabe-Puffer für Betriebssystem */
  unsigned Pos;         /* gewünschte Position des Monitors  */
  unsigned Index;       /* bei Tastatur/Maus: ScreenGroup    */
```

Bild 12: *Die Funktion* `DosMonReg( )`.

Gerätemonitore und das API

Insgesamt sind nur fünf verschiedene Funktionen des API erforderlich, um einen Gerätemonitor zu installieren und um die Datenpakete zu verarbeiten. Zu Programmanfang muß mit der Funktion `DosMonOpen( )` (*Bild 11*) eine Handle für das Gerät, in dessen Datenstrom der Monitor installiert werden soll, angefordert werden. Durch die Anforderung der Handle wird der Gerätetreiber darüber informiert, daß ein Monitor installiert werden soll. Wenn `GeräteName` einen String mit dem logischen Namen eines Gerätes, daß Monitore unterstützt, enthält, erhält das Monitorprogramm über die Adresse `MonHandle` eine Handle, die für weitere Bezüge gesichert werden muß.

Der nächste Schritt besteht darin, den Monitor vom Betriebssystem registrieren und einrichten zu lassen. Dazu wird die Funktion `DosMonReg( )` aufgerufen (*Bild 12*), mit der die vom Betriebssystem benutzten Datenpuffer bereitgestellt und installiert werden. Der Funktion `DosMonReg( )` wird als ein Argument die von der Funktion `DosMonOpen( )` erhaltene Handle übergeben. Zwei weitere Argumente, `InBuf` und `OutBuf`, erhalten die Adressen der Ein- und Ausgabepuffer, die nicht vom Monitorprogramm, sondern vom Betriebssystem genutzt werden, um die Daten an den Monitor zu übertragen und sie vom Monitor an den nächsten Monitor in der Kette weiter- bzw. zum Gerätetreiber zurückzureichen.

Mit dem Argument `Pos` kann festgelegt werden, an welcher Stelle der Monitorkette der neu einzurichtende Monitor plaziert werden soll. Der Monitor kann an die erste oder letzte Position in der Kette installiert werden oder durch die Angabe einer Kennziffer bestimmen, daß die Position keine Rolle spielt. Ein Monitor, der sich an der ersten Stelle der Kette befindet, hat somit auch zuerst die Möglichkeit, das Datenpaket zu erhalten und damit zu modifizieren.

```
DosMonRead(InBuf, WaitFlag, DataBuf, ByteCnt);
  char far *InBuf;     /* Betriebssystem-Puffer              */
  unsigned WaitFlag;  /* Ø= warten, 1 = nicht warten      */
  char far *DataBuf;  /* Puffer für Tastatur-Datenpaket */
  unsigned *ByteCnt;  /* Größe von DataBuf                */
```

Bild 13: *Die Funktion* `DosMonRead( )`.

```
DosMonWrite(OutBuf, DataBuf, ByteCnt);
  char far *OutBuf;   /* System-Ausgang-Puffer  */
  char far *DataBuf;  /* Tastatur-Datenpaket    */
  unsigned ByteCnt;   /* Größe von DataBuf      */
```

Bild 14: *Die Funktion* `DosMonWrite( )`.

```
DosMonClose(MonHandle);
  unsigned MonHandle;    /* Handle, der von DosMonOpen()   */
                         /* zurückgegeben wurde            */
```

Bild 15: *Die Funktion* `DosMonClose( )`.

Die Kontrolle über die Daten, die vom Gerätetreiber bereitgestellt werden, ist somit am direktesten. Der Vorteil bei der Installierung am Ende der Kette besteht darin, daß der sich dort befindende Monitor alle Datenpakete sehen und auch modifizieren kann, die von anderen Monitoren abgeschickt wurden. Auch wenn eine bestimmte Position in der Kette angefordert werden kann, besteht keine Garantie, daß sich das installierte Monitorprogramm auch an der gewünschten Stelle befindet. Darüber hinaus wird der Gerätemonitor auch nicht darüber informiert, an welcher Stelle in der Kette er sich befindet. Das erste Monitorprogramm, daß die Positionierung am Anfang oder Ende der Kette anfordert, wird normalerweise auch dort plaziert. Programme, die später an der gleichen Stelle plaziert werden wollen, erhalten immer die nächste oder (bei Programmen, die am Ende installiert werden sollen) die vorhergehende Position. Das fünfte Argument, `Index`, wird im Zusammenhang mit logischen Geräten benutzt. Da von OS/2 ein Drucker unterstützt wird, sollte dieses Argument bei Druckern den Wert 0 haben. Bei Monitorprogrammen, die die Tastatur oder Maus unterstützen, erhält dieses Argument die Identifikationsnummer der Bildschirmgruppe, für die das Monitorprogramm installiert werden soll. Wenn ein Monitorprogramm im Hintergrund gestartet wird, ist es nicht mit einem bestimmten Vordergrundprozeß verknüpft. Die Verknüpfung des Monitorprogrammes mit einer ScreenGroup wird durch die Angabe der Identifikationsnummer der gewünschten Bildschirmgruppe erreicht. Die zur Ermittlung dieser ID notwendigen Schritte sehen wir uns noch genauer an.

Nachdem das Monitorprogramm erfolgreich installiert ist, kann der Datenstrom, der vom Treiber ausgeschickt wird, überwacht werden. Hierzu wird die Funktion `DosMonRead( )` (*Bild 13*) eingesetzt. Mit dieser Funktion werden die einkommenden Daten gelesen (eigentlich von `InBuf` nach `DataBuf` kopiert) und können nun vom Monitorprogramm untersucht werden.

`DosMonRead( )` werden folgende Argumente übergeben: `InBuf` erhält die Adresse des Eingabepuffers, der auch an die Funktion `DosMonReg( )` übergeben wurde. Über das zweite Argument kann festgelegt werden, ob auf das nächste Datenpaket gewartet, oder, wenn keinen Daten vorliegen, mit einem Fehlercode an die aufrufende Funktion zurückgekehrt werden soll. Liegen Daten vor, kopiert das Betriebssystem die Informationen aus dem Eingabepuffer in den lokalen Puffer des Monitorprogramms, in dem eingekommene Daten untersucht werden können. Das Format von `DataBuf` ist Ihnen ja bereits bekannt. Üblicherweise wird ein Monitorprogramm für die Tastatur auf den Hot-Key warten, mit dem es seine Aktivitäten starten soll.

Der Aufruf der Funktion `DosMonWrite( )` (*Bild 14*) kopiert den Inhalt des lokalen Puffers `DataBuf` in den mit `DosMonReg( )` registrierten Ausgabepuffer `OutBuf`. An dieser Stelle hat das Monitorprogramm die Kontrolle darüber, ob die eingelesene Taste weitergereicht, eine andere Taste in den Datenstrom eingegeben oder auch mehrere Datenpakete in den Datenstrom geschrieben werden.

Typischerweise wird ein weiterer Hot-Key dazu benutzt, um das Monitorprogramm wieder zu deinstallieren. Die teilweise bei MS-DOS auftretenden Probleme bei Deaktivieren eines TSR-Programms, und die Tatsache, das meist nur das zuletzt installierte Programm wieder entfernt werden kann, entfallen bei OS/2 völlig. Durch Aufruf der Funktion `DosMonClose( )` (*Bild 15*) kann zu einem beliebigen Zeitpunkt der Monitor wieder entfernt werden. Diese Funktion erhält als einziges Argument die Handle, die mit der Funktion `DosMonOpen( )` angefordert wurde.

Nachdem die grundlegenden Schritte bei der Programmierung eines Gerätemonitors erläutert worden sind, wollen uns nun besonderen Fragen im Detail zuwenden.

Die Ermittlung der Identifikationsnummer des Vordergrundprozesses

OS/2 verwaltet zwei Informationssegmente, von denen das globale Informationssegment Prozeßinformationen enthält, die für alle derzeit installierten Prozesse gültig und zugänglich sind. Im lokalen Informationssegment sind Informationen enthalten, die spezifisch für einen Prozeß sind, und nur von ihm gelesen werden dürfen.

Zur Installation eines Gerätemonitors für die Tastatur, der ja im Hintergrund ablaufen soll, benötigen wir als Argument für `DosMonReg( )` die Identifikationsnummer der

```
DosGetInfoSeg(GlobalSeg, LocalSeg);
  unsigned *GlobalSeg; /* erhält Selektor globales Segment */
  unsigned *LocalSeg;  /* erhält Selektor lokales Segment  */
```

Bild 16: *Die Funktion* `DosGetInfoSeg( )` *ermittelt die Selektoren für das globale und lokale Informationssegment.*

```
DosSetPrty(Scope, Class, Delta. ID);
  unsigned Scope;   /* legt Einflußbereich fest          */
                    /* Ø: angegebener Prozeß/alle Threads */
                    /* 1: angegebener Prozeß und alle Kind- */
                    /*    prozesse                        */
                    /* 2: ein Thread im aufrufend. Prozeß  */
  unsigned Class;   /* Klasse, die zugewiesen werden soll  */
  unsigned Delta;   /* Veränderung der Stufe (-31 bis +31) */
  unsigned ID;      /* Prozeß-ID, wenn Scope == Ø oder 1;  */
                    /* Thread-ID, wenn Scope == 2          */
```

Bild 17: *Die Funktion* `DosSetPrty( )` *ermöglicht die Veränderung der Priorität eines Prozesses oder Threads.*

ScreenGroup, die sich im Vordergrund befindet. Glücklicherweise können wir dieses Datum dem globalen Informationssegment entnehmen. Wir verwenden dazu die Funktion `DosGetInfoSeg( )`, die uns je einen Selektor für das globale und lokale Informationssegment zurückgibt (*Bild 16*).

In der Funktion `AktiveSGroup( )` leiten wir aus diesen Selektoren Zeiger ab, mit dem wir auf die in der Header-Datei `bsedos.h` deklarierte Struktur `GINFOSEG` zugreifen können. Dem Strukturelement `sgCurrent` (an Offset 18h) entnehmen wir dann die Identifikationsnummer der Bildschirmgruppe, die sich beim Aufruf von BILDOS2 im Vordergrund befindet.

Prioritäten der verschiedenen Monitor-Threads

Wie Sie bereits wissen, können für den Gerätetreiber der Tastatur mehrere Monitorprogramme eingerichtet werden. Da die Informationen über die eingegebenen Tasten, bevor sie die im Vordergrund befindliche Anwendung erreichen, durch alle Monitorprogramme laufen, ist es wichtig, die zur Untersuchung der erhaltenen Daten benutzte Zeit so gering wie möglich zu halten. Wir müssen dafür sorgen, daß der Thread, der die Tastatureingaben untersucht, möglichst viel CPU-Zeit zugeteilt bekommt.

Das Multitasking-Konzept von OS/2 ermöglicht es, für Prozesse oder Threads innerhalb eines Prozesses die Priorität zu verändern. Hier gilt: Je höher die Priorität, desto mehr CPU-Zeit wird dem Prozeß oder Thread von OS/2 zur Verfügung gestellt. Die Priorität läßt sich durch die Zugehörigkeit zu einer Klasse oder Stufe bestimmen. OS/2 kennt drei Prioritätsklassen: zeitkritisch, normal, und faul. Jede Klasse wird außerdem in 32 Stufen (0 - 31) unterteilt.

```
DosAllocShrSeg(Size, Name, Selektor);
  unsigned Size;           /* Größe des Segmentes      */
  char far *Name;          /* Zeiger auf Segmentnamen */
  unsigned far *Selektor   /* Adresse für Selektor     */
```

Bild 18: *Die Funktion* `DosAllocShrSeg( )` *fordert ein gemeinsam benutzbares, benanntes Segment an.*

Mit der API-Funktion `DosSetPrty( )` (*Bild 17*) kann die Priorität von Prozessen verändert werden, wenn in CONFIG.SYS der Befehl `priority=dynamic` steht.

Damit `KbdThread( )` möglichst viel Zeitscheiben erhält, weisen wir ihm die zeitkritische Prioritätsklasse zu, lassen jedoch seine Stufe unverändert.

Mehrfache Installation vermeiden

Eine der goldenen Grundregeln der Programmierung in einer Multitaskingumgebung lautet: »Verwende so wenig Speicherplatz wie möglich«. Aus diesem Grunde ist es wünschenswert, beim Laden eines Monitorprogramms zu überprüfen, ob für die derzeit aktive ScreenGroup (die Bildschirmgruppe, die sich im Vordergrund befindet) der Monitor bereits installiert wurde.

Diese Überprüfung durchzuführen ist ein einfaches Unternehmen: Wir verwenden hierzu den Mechanismus des gemeinsam benutzbaren Datensegments (Shared Segment). Diese Technik wurde eigentlich für den Datenaustausch zwischen verschiedenen Prozessen oder Threads implementiert, eignet sich jedoch für unsere Zwecke hervorragend. Ein Shared Segment wird im Speicher angelegt und erhält einen Pfadnamen, über den es dann ansprechbar ist. Dieser Pfadname muß mit dem Verzeichnisnamen `\SHAREMEM` beginnen, danach kann der Name eines Unterverzeichnisses folgen, der Name des benannten Segments ist vorgeschrieben und folgt den DOS-Konventionen für Dateinamen.

Mit der Funktion `DosAllocShrSeg( )` (*Bild 18*) wird dieses gemeinsame benutzbare Datensegment angefordert. Die Funktion erhält als Argumente die Größe des angeforderten Segmentes, einen Zeiger auf dessen Namen und einen Zeiger, über den die Funktion einen Selektor auf das Segment zurückgibt. Da wir über das Vorhandensein des Shared Segments überprüfen wollen, ob das Monitorprogramm bereits installiert ist, begnügen wir uns mit einer Größe von einem `int`.

Diese API-Funktion gibt den Fehlercode 183 zurück, wenn ein Datensegment gleichen Namens bereits existiert. Somit fällt auf Grund des Rückgabewertes der Funktion die Entscheidung, ob das Programm bereits installiert worden ist oder nicht. Ist dies der Fall, wird das Programm mit einer entsprechenden Meldung, die den Anwender über den Abbruch der Installation informiert, beendet. Anderenfalls wird die Installationsroutine fortgeführt.

```
VioPopUp (WaitFlag, VioHandle);
  unsigned *WaitFlag;  /* Bit 0: 0 = nicht warten,          */
                       /*        1 = auf Bildschirm warten  */
                       /* Bit 1: 0 = nicht transparent      */
                       /*        1 = transparent            */
  unsigned VioHandle;  /* derzeit immer 0                   */
```

Bild 19: *Die Funktion* `VioPopUp( )`.

```
VioEndPopUp(VioHandle);
  unsigned VioHandle;      /* derzeit immer 0        */
```

Bild 20: *Die Funktion* `VioEndPopUp( )`.

Bildschirmausgabe durch einen Hintergrundprozeß

Das letzte Problem, das wir noch lösen müssen, besteht darin, dem im Hintergrund gestarteten Prozeß Zugriff auf den Ausgabebildschirm zu ermöglichen. Dies ist zum einen erforderlich, um die Info-Fenster auszugeben, zum anderen muß das Programm Zugang zum Bildschirm haben, um seinen eigentlichen Zweck, die Erstellung von Dateien mit dem Inhalt des Bildschirms, erfüllen zu können.

Normalerweise hat ein Hintergrundprozeß keinen Bildschirmzugriff. Er kann ihn über die API-Funktion `VioPopUp( )` anfordern (*Bild 19*) und dabei angeben, ob auf das Freiwerden des Bildschirms gewartet werden soll oder nicht. Wenn der Hintergrundprozeß den Bildschirm erhält, sichert das Betriebssystem den aktuellen Zustand des Bildschirms. Eine weitere Option von `VioPopUp( )` legt fest, ob der Bildschirm nach dem Sichern gelöscht werden (nicht transparent) oder unverändert (transparent) bleiben soll.

Wir verwenden an beiden Stellen im Programm, die die Pop-up-Funktionen benutzen - im Thread `BildThread( )` und der Funktion `ZeigeWindow( )` - die Option »warten, nicht löschen«. Den daraus resultierenden Effekt können Sie beobachten, wenn Sie das Programm installieren, in ein Verzeichnis mit einer großen Anzahl von Dateien wechseln, den DIR-Befehl eingeben und mit [Alt][Enter] BILDOS2 den Befehl zum Speichern des Bildschirms geben. BILDOS2 erhält erst dann Zugriff auf den Bildschirm, wenn der DIR-Befehl beendet wurde. Während des Popups können die üblichen Ausgabefunktionen des Vio-Moduls benutzt werden.

Der temporäre Bildschirmzugriff wird über die Funktion `VioEndPopUp( )` wieder freigegeben (*Bild 20*) - was so schnell wie möglich passieren sollte. Eine Einschränkung der Pop-up-Funktion besteht darin, daß immer nur ein Pop-up-Bildschirm zur gleichen Zeit aktiv sein kann. Wenn ein anderes Programm diesen temporären Bildschirm besitzt, muß das weitere anfordernde Programm so lange warten, bis der andere Prozeß diesen wieder freigegeben hat.

Die Realisierung der Hardcopy-Funktion

Die mit BILDOS2 erstellten Bildschirmhardcopies sind für den 80x25 Textbildschirm immer 4000 Bytes groß, da sowohl sämtliche Zeichen, als auch die zugehörigen Attribute des Bildschirms gesichert werden können. Dieses Format wurde gewählt, damit mit dem in der März/April-Ausgabe von MSJ beschriebenen Programm BW.EXE Dateien erstellt werden können, in denen die erforderlichen Attribute des Textes im Word/Datei-Format enthalten sind. Wurde entdeckt, daß der Aktivierungs-Hotkey eingegeben wurde, wird ein Thread gestartet, der die 2000 Zeichen mit ihren Attributen in einen dynamisch allokierten Datenpuffer ablegt. Wurde der Bildschirminhalt einmal gesichert, kann das Vordergrund-Programm weiterlaufen. Dieser 4 Kbyte große Puffer wird nach dem Sichern des Bildschirminhalts in der Datei wieder freigegeben.

Mit der Funktion `DosOpen( )` wird die Datei, in die der Bildschirminhalt geschrieben werden soll, geöffnet, die Daten mit `DosWrite( )` weggeschrieben und abschließend mit `DosClose( )` die Datei wieder geschlossen.

Rainer G. Haselier

```
/**************************************************************/
/* Datei:              bildos2.c                              */
/**************************************************************/
/* Version:            1.03                                   */
/* entwickelt:         12. 11. 1988                           */
/* letzte Änderung:     3.  3. 1989                           */
/* Aufbau:             cl -AS -Lp -Zp -G2 -W3 bildos2.c       */
/* Start mit:          detach bildos2                         */
/**--------------------------------------------------------**/
/*                                                            */
/* Beschreibung:   Gerätemonitor für OS/2, mit dem Hardcopy- */
/*                 Dateien von Text-Bildschirmen im Modus    */
/*                 80 Spalt./25 Zeilen erstellt werden können*/
/*                 Mit dem Programm bw.exe (siehe MSJ März/  */
/*                 April 89) können die Dateien in das Word- */
/*                 Dateiformat umgewandelt werden.           */
/*                                                            */
/*                 Die Hardcopy-Dateien werden in das akt.   */
/*                 Verzeichnis geschrieben und erhalten den  */
/*                 Namen BILDXXXX.BLD, wobei die Bilder bei  */
/*                 0 beginnend durchnumeriert werden.        */
/*                                                            */
/**************************************************************/
/*    (c) copyright 1988,1989 rabbitsoft Rainer G. Haselier  */
/*                 Alle Rechte vorbehalten.                   */
/**************************************************************/

/**************************************************************/
/*                      INCLUDE-DATEIEN                       */
/**************************************************************/
#define INCL_DOS
#define INCL_SUB
#include <os2.h>              /* API-Prototypen/Strukturen   */
#include <stdio.h>            /* für NULL, sprintf(), ...    */
#include <string.h>           /* für strlen()                */
#include <malloc.h>           /* für malloc()                */
```

Listing 1: `BILDOS2.C`

```
/**************************************************************/
/*                SYMBOLISCHE KONSTANTEN                      */
/**************************************************************/
#define POS_EGAL   0             /* mögl. Argumente DosMonReg(),*/
#define POS_VORNE  1             /* legt Positionswunsch für    */
#define POS_HINTEN 2             /*      Monitor fest, hier: 1  */

#define WAIT 0                   /* Monitor wartet auf Eingabe  */

                                 /* Hotkeys festlegen           */
#define HOT_ASCII    0           /* Taste mit erweitertert. Code*/
#define HOT_SCAN_1  28           /* Alt+Enter macht Bild        */
#define HOT_SCAN_2  14           /* Alt+Backspace Ende Monitor  */
#define RELEASE     0x40         /* Taste losgelassen           */

                                 /* für Datei-Operationen       */
#define ATTR 0                   /* normales Datei-Attribut     */
#define OPENFLAG 0x12            /* neue Datei, oder ersetzen   */
#define OPENMODE 0x0041          /* nur schreiben/nicht vererben*/

#define BYTE_PRO_BILD 4000       /* für DosWrite(): Dateigröße  */

#define SEGMENT_SCHON_DA 183     /* benutzter API-Fehlercode    */

#define VIOHANDLE 0              /* derzeit immer 0             */

#define STACKGROESSE 2048        /* Stackgröße neue Threads     */


/**************************************************************/
/*                DATENSTRUKTUREN                             */
/**************************************************************/
typedef struct tag_KeyPacket{  /* Struktur Tastatur-Datenpack*/
   unsigned        monflags;            /* Monitor Flag-Word  */
   unsigned char   ascii;               /* Taste: ASCII-Code  */
   unsigned char   scan;                /* Taste: Scan-Code   */
   unsigned        nls;                 /* Sprachensupport    */
   unsigned        shift;               /* Status Shift-Tasten*/
   unsigned long   keytime;             /* Zeit Tastendruck   */
   unsigned        ddflags;             /* Flags Gerätetreiber*/
} KEYPACKET;


typedef struct tag_MonBuff {   /* Struktur Ein-/Ausgabepuffer*/
   unsigned  length;                    /* Länge von MONBUFF  */
   unsigned char buffer[128];           /* vom System benutzt */
} MONBUFF;

/**************************************************************/
/*                GLOBALE DATEN                               */
/**************************************************************/
MONBUFF MonInBuf, MonOutBuf;   /* Datentransfer-Puffer        */
KEYPACKET kp;                  /* Tastatur-Datenpaket         */

unsigned long SemPrgEnde = 0L; /* Semaphor frei=Programm-Ende*/
unsigned long SemBild    = 0L; /* Semaphor frei=Bild machen   */

char cAttr = 0x70;             /* Attribut für Info-Fenster   */
unsigned BildNr = 0;           /* Zähler für Bildnummer       */

/**************************************************************/
/*                INFO-FENSTER                                */
/**************************************************************/
char *apstrAktiv[] =
{
   "                                                    ",
   "  BildOS2, Version 1.03                            ",
   "  (c) copyright 1989 rabbitsoft, Rainer G. Haselier.",
   "                                                    ",
   "    Bild erstellen:     [Alt]+[Enter]              ",
   "    Monitor beenden:    [Alt]+[Backspace]          ",
   "                                                    ",
   "  weiter mit beliebiger Taste...                   ",
   "                                                    ",
```

Listing 1: *(Fortsetzung)*

```
   NULL
};

char *apstrDeAktiv[] =
{
   "                                                    ",
   "           BildOS2, Version 1.03                   ",
   "   Programm wurde erfolgreich deinstalliert.       ",
   "                                                    ",
   NULL
};

char *apstrSchonDa[] =
{
   "                                                    ",
   "           BildOS2, Version 1.03                   ",
   " Programm in dieser ScreenGroup bereits installiert ",
   "                                                    ",
   NULL
};

char *apstrFehler[] =
{
   "                                                    ",
   "           BildOS2, Version 1.03                   ",
   "   Programm konnte nicht installiert werden.       ",
   "                                                    ",
   NULL
};


/**************************************************************/
/*                PROTOTYPEN                                  */
/**************************************************************/
void main(void);
void ZeigeWindow(int, int, int, char*[], char*);
void FehlerEnde(void);
void far BildThread(void);
void far KbdThread(void);
unsigned AktiveSGroup(void);


/**************************************************************/
/*                MAIN-FUNKTION                               */
/**************************************************************/
void main(void)
{
   unsigned short MonHandle;             /* Handle Monitor      */
   unsigned sgCurID,                     /* ID ScreenGroup      */
            ret;                         /* Rückgabewert API    */

   unsigned short SharSegSel;            /* Selektor und Name   */
   char SharSegName[30];                 /* des Shared Segment  */

   char far *bildStack;                  /* Stack und ID des    */
   unsigned bildID;                      /* Bild-Thread         */

   char far *kbdStack;                   /* Stack und ID des    */
   unsigned kbdID;                       /* Tastatur-Threads    */


                                         /* Stack allokieren    */
   bildStack = (char far*) malloc(STACKGROESSE) + STACKGROESSE;
   kbdStack  = (char far*) malloc(STACKGROESSE) + STACKGROESSE;

   sgCurID = AktiveSGroup();             /* ID aktive Screen-   */
                                         /*   Group ermitteln   */

   sprintf(SharSegName,                  /* Name des Shared     */
           "\\SHAREMEM\\BILDOS2.%02u",   /*  Segment festlegen*/
           sgCurID);
   ret = DosAllocShrSeg(sizeof(int),     /* Segment anfordern   */
            SharSegName, &SharSegSel);
```

Listing 1: *(Fortsetzung)*

```
    if (ret == SEGMENT_SCHON_DA)           /* wenn vorhanden       */
    {                                      /* Meldung ausgeben     */
       ZeigeWindow(8, 10, 2, apstrSchonDa, &cAttr);
       DosExit(1,1);                       /* und beenden          */
    }
    if (DosSemSet(&SemPrgEnde))            /* Semaphore belegen    */
       FehlerEnde();
    if (DosSemSet(&SemBild))
       FehlerEnde();
                                           /* Größe der Puffer     */
                                           /*     eintragen        */
    MonInBuf.length  = (unsigned) sizeof( MONBUFF);
    MonOutBuf.length = (unsigned) sizeof( MONBUFF);

    if (DosMonOpen("KBD$", &MonHandle)) /* Verbindung zum      */
       FehlerEnde();                       /* Treiber herstellen */
    if (DosMonReg( MonHandle,              /* Monitor registrier.*/
          (unsigned char *) &MonInBuf,     /*     lassen          */
          (unsigned char *) &MonOutBuf,
          POS_VORNE, sgCurID))
       FehlerEnde();

    if (DosCreateThread(KbdThread,         /* KbdThread erzeugen */
          &kbdID, kbdStack))
       FehlerEnde();

    if (DosSetPrty(PRTYS_THREAD,           /* und dessen          */
          PRTYC_TIMECRITICAL, 0, kbdID))/* Priorität erhöhen  */
       FehlerEnde();

    if (DosCreateThread(BildThread,        /* BildThread erzeugen*/
          &bildID, bildStack))
       FehlerEnde();

    ZeigeWindow(7, 13, 0,                  /* Installationsmel-  */
       apstrAktiv, &cAttr);                /*    dung ausgeben   */

    if (DosSemWait(&SemPrgEnde, -1L))      /* warten bis Semaphor*/
       FehlerEnde();                       /*    freigegeben wird*/

                                           /* ROUTINEN ZUM BEEN- */
                                           /* DEN DES MONITORS   */
    DosSuspendThread(kbdID);               /* laufende Threads   */
    DosSuspendThread(bildID);              /*      unterbrechen  */

    ZeigeWindow(10, 10, 1,                 /* Deinstallationsmel-*/
       apstrDeAktiv, &cAttr);              /*     dung ausgeben  */

    DosFreeSeg(SharSegSel);                /* Segment freigeben  */
    DosMonClose(MonHandle);                /* Monitor schliessen */
    DosExit ( EXIT_PROCESS, 0 );           /* zurück zu OS/2     */
}

/************************************************************
* Funktion:      AktiveSGroup                               *
**--------------------------------------------------------**
* Parameter:     keine                                      *
* Rückgabewert:  unsigned     ID der Vodergrund-ScreenGroup *
* Globale Daten: keine                                      *
*----------------------------------------------------------*
* Beschreibung:  Diese Funktion ermittelt die ID der Screen- *
*                Group, die sich derzeit im Vordergrund befin-*
*                det. Diese ID wird für die Festlegung des  *
*                Namens des Shared Segments und zum Registrie-*
*                ren des Monitors benötigt.                 *
************************************************************/
unsigned AktiveSGroup(void)
{
    unsigned short infoseg, localseg;
    struct _GINFOSEG far *globptr;

    if (DosGetInfoSeg(&infoseg, &localseg))
       FehlerEnde();
    globptr = (struct _GINFOSEG far*) ( (long) infoseg<<16);

    return (globptr->sgCurrent);
}
```

Listing 1: *(Fortsetzung)*

```
/************************************************************
*                                                           *
* Thread:        KbdThread                                  *
*                                                           *
**--------------------------------------------------------**
* Parameter:     keine                                      *
* Rückgabewert:  keiner                                     *
* Globale Daten: SemBild      (W - indirekt über API)       *
*                SemPrgEnde   (W - indirekt über API)       *
*                MonInBuf     (W - indirekt über API)       *
*                MonOutBuf    (W - indirekt über API)       *
*                kp           (R - indirekt über API)       *
*----------------------------------------------------------*
* Beschreibung:  Dieser Thread ist die eigentliche Über-    *
*                wachungsfunktion für die Tastatureingaben. *
*                Hier wird überprüft, ob die Hotkeys zum Ab- *
*                speichern des Bildes oder zum Beenden des  *
*                Programms eingegeben wurden.               *
*                Soll ein Bild abgespeichert werden, wird das *
*                Semaphor SemBild freigegeben, und damit der *
*                Thread BildThread() gestartet. Wurde der Hot-*
*                key zum Beenden des Programmes entdeckt, wird*
*                das Semaphor SemPrgEnde freigegeben, wodurch *
*                die Deinstallierungsfunktionen in main()   *
*                abgearbeitet werden können.                *
*                Alle anderen Daten werden unverändert an den *
*                nächsten Monitor weiter-, oder den Geräte- *
*                treiber zurückgegeben.                     *
************************************************************/
void far KbdThread(void)
{
    unsigned ByteCnt;

    for (;;)
    {
       ByteCnt = sizeof(kp);               /* Datenpaket lesen    */
       DosMonRead( (unsigned char*) &MonInBuf,
          WAIT, (char *) &kp, &ByteCnt);

       if((kp.ddflags && RELEASE) == 0) /* Taste losgelassen  */
       {
          if( (kp.ascii == HOT_ASCII)&& /* Bild erstellen      */
              (kp.scan  == HOT_SCAN_1))
             DosSemClear(&SemBild);        /* Semaphor freigeben */
                                           /* Programm beenden   */
          else if( (kp.ascii== HOT_ASCII) &&
                   (kp.scan == HOT_SCAN_2))
             DosSemClear(&SemPrgEnde);
       }

                                           /* Daten weiterreichen*/
       DosMonWrite( (unsigned char*) &MonOutBuf,
                    (char*) &kp, ByteCnt);
    }
}


/************************************************************
*                                                           *
* Thread:        BildThread                                 *
*                                                           *
**--------------------------------------------------------**
* Parameter:     keine                                      *
* Rückgabewert:  keiner                                     *
* Globale Daten: SemBild (R/W), BildNr (R/W)                *
*----------------------------------------------------------*
* Beschreibung:  Dieser Thread wird durch die Freigabe des  *
*                RAM-Semaphors SemBild zum Leben erweckt.   *
*                Mit den API-Popup-Funktionen erhält die    *
*                Funktion Zugang zum Bildschirm der Vorder- *
*                grund-Screengroup, liest den Bildschirmin- *
*                halt, und schreibt ihn in eine Datei.      *
*                Tritt beim Öffnen der Datei ein Fehler auf, *
*                werden zwei Warntöne ausgegeben.           *
************************************************************/
```

Listing 1: *(Fortsetzung)*

```
void far BildThread(void)
{
   char far *Puffer = NULL;                /* Zeiger auf Puffer  */
   SEL Selektor;                           /* Selektor auf Puffer*/
   HFILE hfDatei;                          /* Datei Handle       */
   unsigned NoClear = Øx03;                /* warten, transparent*/
   unsigned Aktion;                        /* Aktion beim Öffnen */
   unsigned Geschrieben;                   /* geschriebene Bytes */
   unsigned ByteProBild = BYTE_PRO_BILD;
   unsigned long Groesse = Ø;              /* Größe der Datei    */
   unsigned char acDateiname[128];         /* erhält Dateinamen  */
   for (;;)
   {
      DosSemWait( &SemBild, -1L );         /* Warte, bis Semaphor*/
                                           /* frei wird          */
      DosAllocSeg( ByteProBild,            /* Puffer anfordern   */
                   &Selektor, Ø) ;         /* und Zeiger ableiten*/
      Puffer = (char far*) (((long) Selektor) <<16);


      VioPopUp(&NoClear, VIOHANDLE);       /* Schirm anfordern   */
      VioReadCellStr( Puffer,              /* Inhalt lesen       */
                      &ByteProBild,
                      Ø, Ø, VIOHANDLE );
      VioEndPopUp(VIOHANDLE);              /* PopUp freigeben    */
      sprintf( acDateiname,                /* Dateinamen erzeugen*/
               "BILD%Ø3u.BLD", BildNr);
      if( DosOpen( acDateiname,            /* Datei öffnen       */
                   &hfDatei, &Aktion,
                   Groesse, ATTR,
                   OPENFLAG, OPENMODE, ØL) )
      {                                    /* im Fehlerfall      */
         DosBeep(1ØØØ, 5ØØ);               /*     zweimal beepen */
         DosBeep(1ØØØ, 5ØØ);
      }
      else                                 /* sonst              */
      {                                    /* Pufferinhalt in    */
         DosWrite( hfDatei, Puffer,        /* Datei wegschreiben */
                   BYTE_PRO_BILD, &Geschrieben );
         BildNr++;                         /* Bildnummer erhöhen */
      }
      DosClose( hfDatei );                 /* Datei schließen    */
      DosFreeSeg( Selektor );              /* Speicher freigeben */
      DosSemSet( &SemBild );               /* Semaphor belegen   */
   } /* end of for(;;) */

}

/***************************************************************
* Funktion:      ZeigeWindow                                   *
**-----------------------------------------------------------**
* Parameter:     int  Zeile       Zeile des Fensters           *
*                int  Spalte      Spalte des Fensters          *
*                int  Modus       == Ø nach Ausgabe auf bel.   *
*                                       Taste warten           *
*                                 > Ø Pause in Sekunden nach*
*                                       Ausgabe des Fensters   *
*                char* apstrTxt[] Ausgabetext                  *
*                char* pbAttr     Adresse des Attributes       *
* Rückgabewert:  keiner                                        *
* Globale Daten: keine                                         *
```

Listing 1: *(Fortsetzung)*

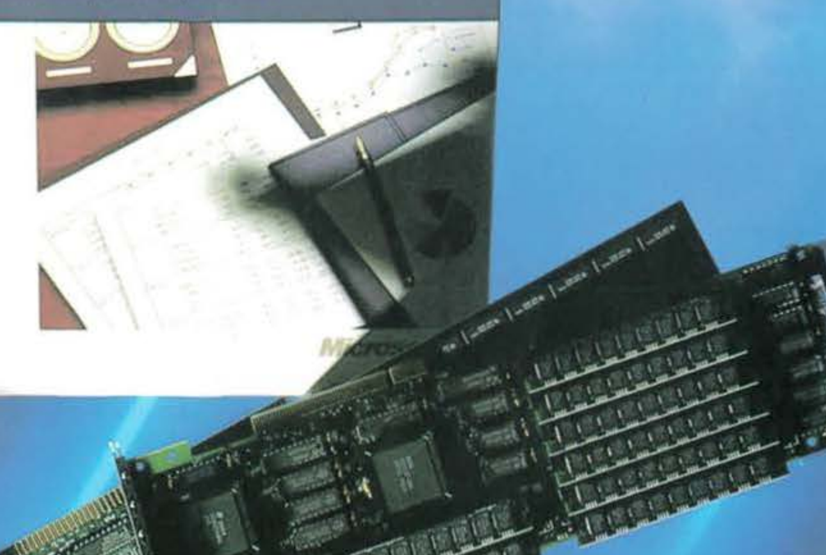
```
*-------------------------------------------------------------*
* Beschreibung:  Diese Funktion gibt ab der übergebenen        *
*                Position (Zeile, Spalte) ein Textwindow aus,  *
*                auf dessen ersten String apstrTxt zeigt.      *
*                Alle Strings des Arrays müssen die gleiche    *
*                Länge besitzen. Zur Ausgabe wird das          *
*                Attribut, auf das pbAttr zeigt, verwendet.    *
*                Über das Argument Modus kann gesteuert wer-   *
*                den, wie das Fenster entfernt wird. Hat       *
*                Modus den Wert Ø, wird auf eine beliebige     *
*                Taste gewartet, anderenfalls wird der Wert    *
*                von Modus als Argument für DosSleep()         *
*                benutzt, und die übergebene Zeit in Sekunden *
*                gewartet, bevor der Ursprungsbildschirm       *
*                wieder hergestellt wird.                      *
***************************************************************/
void ZeigeWindow( int Zeile, int Spalte, int Modus,
                  char* apstrTxt[], char* pbAttr)
{
   KBDKEYINFO Taste;                       /* für KbdCharIn()    */
   int laenge;                             /* Länge des Textes   */
   unsigned WaitNoClear = Øx03;            /* warten, transparent*/
   char **ppstrTmp;                        /* temp. Zeiger String*/

   ppstrTmp = apstrTxt;                    /* Adresse zuweisen   */
   VioPopUp(&WaitNoClear, VIOHANDLE);      /* PopUp anmelden     */
   laenge = strlen(*ppstrTmp);             /* Länge ermitteln    */
   if (laenge != Ø)                        /* wenn Text da       */
   {
      while(*ppstrTmp != NULL)             /* solange Text da,   */
         VioWrtCharStrAtt(*ppstrTmp++,     /*     ausgeben       */
             laenge, Zeile++,              /*     wie Text da    */
             Spalte, pbAttr, VIOHANDLE);
      if (Modus == Ø)                      /* auf beliebige      */
         KbdCharIn(&Taste, Ø, Ø);          /*      Taste warten  */
      else                                 /* sonst übergebene   */
         DosSleep(Modus * 1ØØØL);          /*   Pause einhalten*/
   }
   VioEndPopUp(VIOHANDLE);                 /* PopUp freigeben    */
   return;
}


/***************************************************************
* Funktion:      FehlerEnde                                    *
**-----------------------------------------------------------**
* Parameter:     keine                                         *
* Rückgabewert:  Fehlercode 1 an OS/2 über DosExit()           *
* Globale Daten: apstrFehler (R)                               *
*-------------------------------------------------------------*
* Beschreibung:  Diese Funktion beendet das Programm mit einer*
*                Meldung, wenn bei der Installation unerwar-   *
*                tete Probleme auftreten.                      *
***************************************************************/
void FehlerEnde(void)
{
   ZeigeWindow(7, 13, Ø,                   /* Meldung ausgeben   */
      apstrFehler, &cAttr);
   DosExit(EXIT_PROCESS, 1 );              /* zurück zu OS/2     */
}
```

Listing 1: *(Ende)*

Above™ Board Plus
intel
Word
Microsoft Excel
Vollständiges Tabellenkalkulationsprogramm mit Grafik und Datenbank



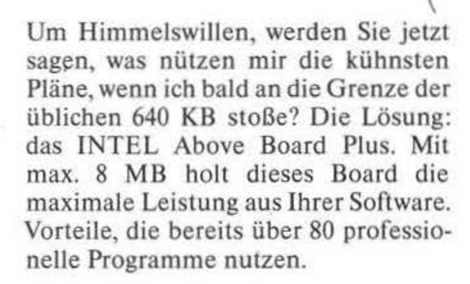
Ein Geschenk des Himmels!

ZWEITER TEIL

Jetzt öffnen sich wieder die Himmelspforten. Ab sofort gibt es das neue Intel Above Board Plus zweimal zum himmlisch günstigen Preis. Das erste heißt Management-Package, besteht aus Above Board Plus, Microsoft Excel und einer Microsoft-Maus.
Das zweite ist das Windows/286-Package und enthält das Board inclusive dem Microsoft Windows/286.
Beide Kombinationen bringen Ihnen den Himmel auf Erden: Zeit- und Geldersparnis durch zwei Pakete, die teuflisch gut zusammen passen. Von zwei führenden Unternehmen der Branche.

1. Das Management-Package

Für die Entscheider von morgen heißt es nicht mehr nur, Fakten und Zahlen zu sammeln. Es gilt, Trends aufzuspüren und Analysen zu erstellen. Dafür brauchen Entscheider das beste Werkzeug. Sie brauchen das Management-Package. Excel nutzt erstmals alle Vorteile von Windows, erleichtert so den späteren Übergang zu OS/2 und übernimmt mühelos alle Daten aus Lotus, dBase, Multiplan etc. Excel vereint Kalkulation, Grafik und Datenbank: So werden aus Zahlen Strategien und aus Informationen Entscheidungen. Und weil die neue Microsoft-Maus serienmäßig zum Package dazugehört, wird nicht nur die Bedienung noch einfacher, sondern das Paket auch himmlisch günstig:
Sie sparen 435,- DM.



**Above Board Plus.
Mehr Raum für kühne Pläne.**

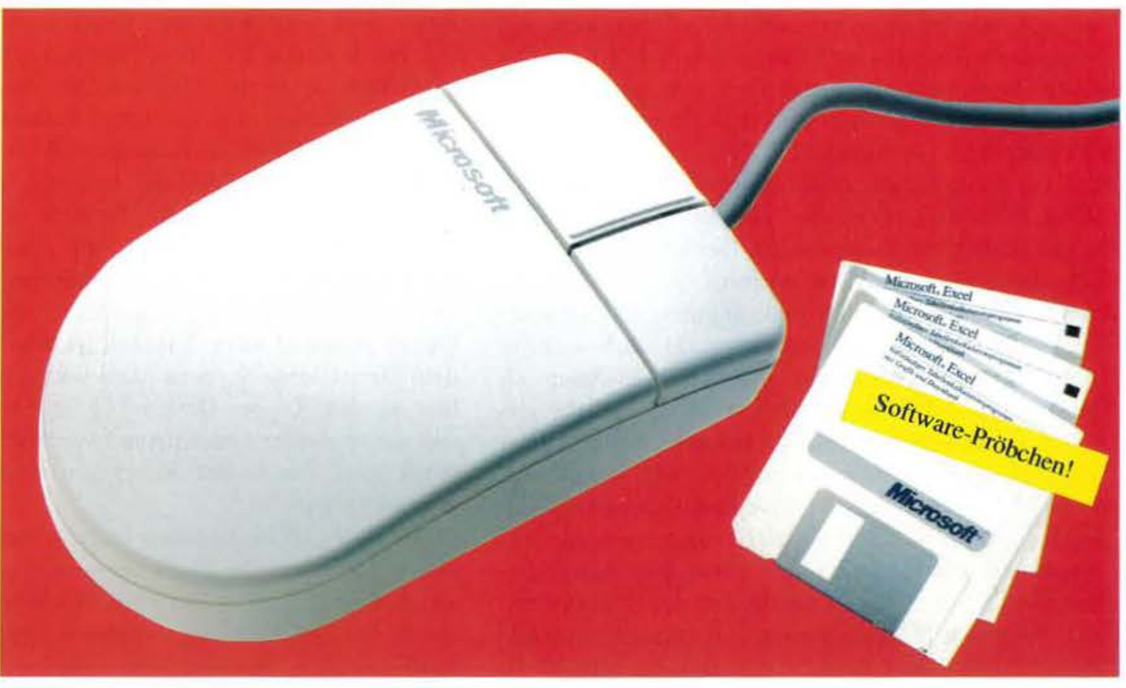
Um Himmelswillen, werden Sie jetzt sagen, was nützen mir die kühnsten Pläne, wenn ich bald an die Grenze der üblichen 640 KB stoße? Die Lösung: das INTEL Above Board Plus. Mit max. 8 MB holt dieses Board die maximale Leistung aus Ihrer Software. Vorteile, die bereits über 80 professionelle Programme nutzen.

2. Das Windows/286-Package

Sie brauchen mehr Speicher, setzen auf INTEL-Qualität und hätten gern Multitasking unter WINDOWS/286? Unser zweites himmlisches Erlebnis, die neuen Above Board Plus Hardware-Register für EMS 4.0 machen all dies möglich: Ab sofort gibt es zu jedem Above Board Plus das neue Windows/286. Und 5 Jahre Garantie.
Sie sparen 450,-DM.

Schauen Sie also schnell bei Ihrem Händler vorbei. Die Angebote gelten, solange der Vorrat reicht. Und es wäre doch die Hölle, würden Sie zu spät kommen. Wir nennen Ihnen gerne einen Fachhändler in Ihrer Nähe.

COMPUTER 2000 AG
Baierbrunner Str. 31
8000 München 70
Tel. (089) 76990-0





Microsoft

COMPUTER 2000
Wir wissen, was läuft.

Informationen aus erster Hand:

Die I/O-Subsysteme von OS/2

OS/2 enthält eine größere Menge von I/O-Services für jene Devices (speziell Tastatur, Videodisplay und Maus), die bei der Implementierung einer OS/2-Benutzerschnittstelle besonders wichtig sind. Ed Iacobucci, der Chefentwickler für OS/2 bei IBM, beschreibt in diesem Artikel diese I/O-Subsysteme.

System-I/O der unteren Stufe wird für diese Devices von speziellen OS/2-Komponenten, den I/O-Subsystemen, durchgeführt. Ein I/O-Subsystem führt die Grund-I/O-Operationen für OS/2-Programme durch. Betrachten Sie diese Subsysteme als Pendants des Protected-Mode zu den BIOS-Services in der PC-DOS-Umgebung. OS/2 implementiert I/O-Subsysteme für die Tastatur, das Videodisplay und die Maus. Jeglicher Input von oder Output für diese Devices (ob sie nun von einer Applikation oder dem System selber stammen) wird am Ende in eine I/O-Subsystem-Primitive aufgelöst.

Warum Subsysteme notwendig sind

Sie mögen sich fragen, warum diese Services existieren. Bietet das Dateimodell nicht alle Funktionen, die für den Zugriff auf diese Devices erforderlich sind? Die Antwort liegt in einem einfachen Kompromiß zwischen Funktion und Ansprechbarkeit. Betrachten Sie, wie diese Kompromisse zum Beispiel bei den Video-Services aussehen.

Das Datei-I/O-Modell kann für Konsolen-I/O eingesetzt werden, der auf Handles basiert. Dieses Modell ist mehr als ausreichend für »Standard«-I/O-Funktionen, wie das Schreiben zur Konsole im Teletype-Mode (TTY). TTY-I/O stimmt tatsächlich genau mit dem Datenstrommodell für serielle Zeichen überein und ist deshalb mit den OS/2-Datei-I/O-Services kompatibel. Die System-Utilities schreiben zur Konsole, als ob die Konsole ein TTY-Device wäre. Dies bietet offensichtliche Vorteile, da OS/2 dann den Output zu anderen Programmen, Devices oder Dateien über Pipes schicken oder umleiten kann.

In der Umgebung des Personalcomputers ist einfaches TTY-I/O nicht immer angemessen. Hochentwickelte Benutzerschnittstellen verlangen, daß der Bildschirm als eine Darstellungsfläche mit wahlfreiem Zugriff auf die verschiedenen Bildschirmkoordinaten behandelt wird. Dies wird ganz klar, wenn man betrachtet, wie »Ganzbildschirm«-Applikationen wie Tabellenkalkulationen den Bildschirm aktualisieren. Ein Spreadsheet-Programm verschiebt ein Viewport (den Bildschirm) über einen sehr viel größeren Darstellungsraum (den Arbeitsraum der Tabellenkalkulation). Teile des Bildschirms werden aktualisiert, wenn der Anwender von einem Bereich zum anderen geht. Es ist keine praktikable Lösung, den gesamten Bildschirm jedesmal, wenn der Anwender den Cursor bewegt, neu zu schreiben.

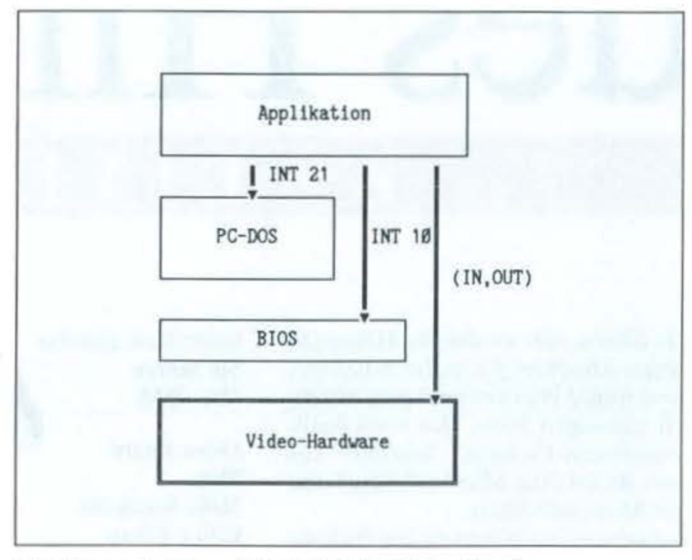


Abbildung 1: *Video-I/O in PC-DOS-Applikationen.*

Außerdem macht I/O-Umleitung in dieser Umgebung keinen Sinn, da die Schnittstelle visuell ist. Die Bedeutung einer jeweilig gedrückten Taste oder eines eingegebenen Zeichens wird nur durch die relative Position des Cursors innerhalb des Spreadsheets bestimmt.

Die meisten Ganzbildschirm-Applikationen erfordern optimale Video-Leistung, da die Antwortzeit der Schnittstelle wichtig für die Verwendbarkeit der Applikation ist. In der PC-DOS-Umgebung werden Ganzbildschirm-Applikationsprogramme selten direkt zur INT21-READ/WRITE-Schnittstelle geschrieben. Statt dessen bevorzugen die Verfasser von Applikationen BIOS-Services oder direkten Hardware-Zugriff (Memory-Mapped-I/O).

Gehen Sie in der funktionellen Hierarchie tiefer nach unten, gewinnen Sie Leistung, verlieren aber Funktion. *Abb. 1* zeigt, wie dieses Konzept sich im Fall von PC-DOS-Video-I/O auswirkt. Auf der höchsten Stufe greifen Applikationsprogramme auf die Konsole über das PC-DOS-Dateisystem zu. In diesem Fall erhält die Applikation Umleitungsdienste, ANSI-Unterstützung und eine Darstellung des Bildschirms auf hoher Stufe, als ob die Konsole ein Dumb-Terminal wäre. Dumb-Terminal ist ein Begriff, mit dem Darstellungs-Services ohne lokale Intelligenz beschrieben werden. Dieser Device-Typ ist häufig so konstruiert, daß ein spezieller Datenstrom (wie ANSI, 3270 etc.) unterstützt wird, der in der Regel von einem Host-Computer gesendet wurde.

Ist Datei-I/O-Umleitung nicht wichtig, kann die Applikation direkt auf die BIOS-Services zugreifen, um wahlfrei Zeichen und Zeichen-Attribute zu schreiben. Diese Technik bietet bessere Leistung, während gleichzeitig Hardwareunabhängigkeit bewahrt und Display-Adapter-Typ, Bildschirmmodus (Grafik oder Text) und charakteristische Eigenschaften des Displays von der Applikation abgeschirmt werden.

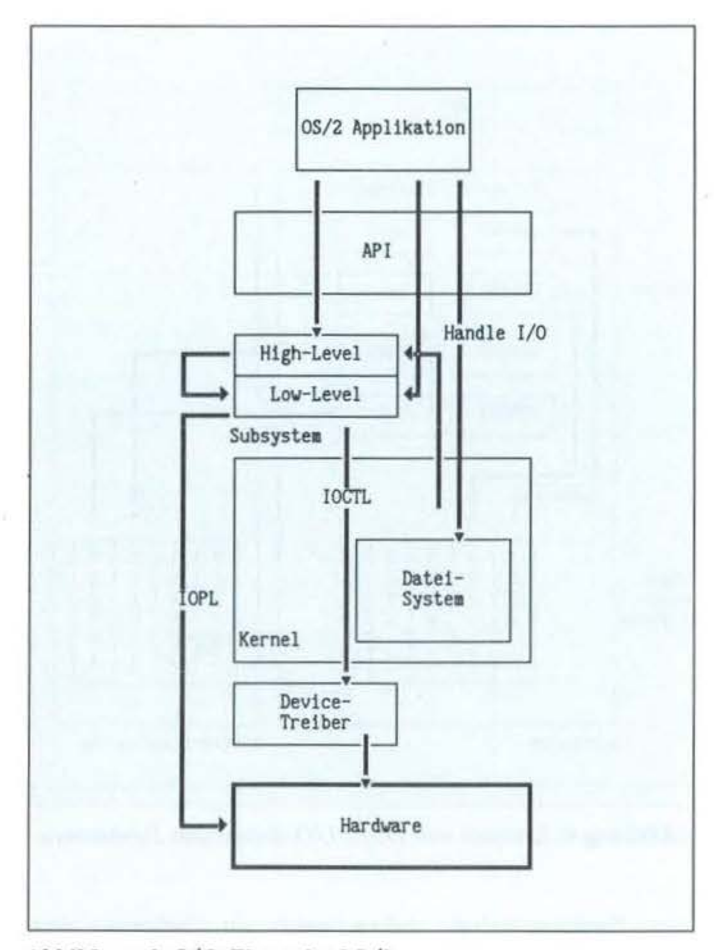


Abbildung 2: *I/O-Flüsse in OS/2.*

Am anderen Ende des Spektrums wird höchste Videoleistung dadurch erreicht, daß der Videodisplay-Buffer direkt beschrieben wird. In diesem Fall muß die Applikation jedoch alle Charakteristika des Devices verstehen, da sie alle Verpflichtungen der System-Videotreiber übernimmt. Anwender, die die Hardware direkt ansprechen, sind mit allen Eigenarten einzelner Display-Adapter vertraut (zum Beispiel mit CGA-Retrace-Synchronisation, um »Schnee« beim Schreiben von Zeichen zu vermeiden).

OS/2-I/O-Subsysteme bieten sich Applikationsprogrammen an, die auf die Attribute des Standard-I/O zugunsten verbesserter Leistung verzichten wollen. Zahlreiche Stufen von Subsystem-Services bieten eine breite Auswahl an Schnittstellenstufen (von TTY-ähnlichem Output mit voller ANSI-Unterstützung bis zur direkten Hardware-Manipulation).

I/O-Struktur von OS/2

Ehe wir uns die einzelnen Subsysteme ansehen, soll ihre allgemeine Implementierungs-Architektur betrachtet werden. Die I/O-Subsysteme werden mit dem OS/2-Subsystemmodell implementiert. Jedes Subsystem bietet einen Satz dynamischer Link-APIs, über die eine Applikation auf seine Services zugreifen kann. Jedes Subsystem besitzt eine entsprechende DLL-Runtime-Datei und einen Device-Treiber. Subsystem-Serviceroutinen werden für das aufrufende Programm als Subroutine auf PL3 im Kontext (LDT) des aufrufenden Programms ausgeführt.

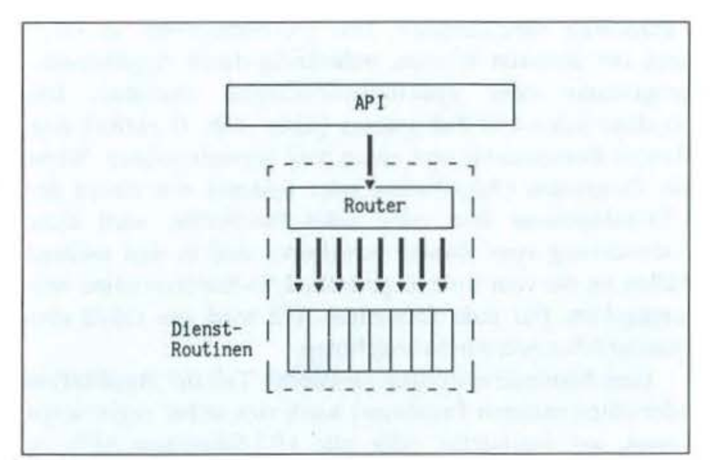


Abbildung 3: *I/O-Subsystem-Struktur.*

Abb. 2 zeigt den I/O-Fluß im Gesamtsystem. Auf der höchsten Stufe kann ein Applikationsprogramm direkte Aufrufe an das Dateisystem oder das I/O-Subsystem ausführen. Im Fall von Handle-Device-I/O geht der Aufruf zuerst zum Dateisystem-Router, damit festgestellt wird, ob Umleitung vorliegt. Ist das Device umgeleitet, wird der Output zum entsprechenden Device oder zur entsprechenden Datei geschickt. Wird das Device nicht umgeleitet, wird die Anforderung an ein I/O-Subsystem zur Erledigung übergeben. In jedem Fall wird die Anforderung am Ende vom Subsystem bedient. Die I/O-Subsysteme in OS/2 bilden die alleinige Schnittstelle zur Tastatur, dem Videodisplay und den Maus-Devices.

Benötigt ein Applikationsprogramm keine Umleitungs-Services (oder wenn es auf das Device wahlfrei zugreifen will), kann das I/O-Subsystem direkt aufgerufen werden. Die I/O-Subsystem-APIs sind ein anderer Teil des Gesamt-OS/2-API. Wenn das Applikationsprogramm diese APIs direkt verwendet, verwirkt es das serielle Datenstrommodell und seine Umleitungsfähigkeiten, gewinnt aber mehr Kontrolle über das Device.

Die I/O-Subsystem API-Aufrufe sind stufig strukturiert, von den eher primitiven Funktionen bis zu höheren Funktionen. Im Fall von Video-I/O werden die grundlegenden TTY-Fähigkeiten durch eine höhere WriteTTY-Schnittstelle ohne I/O-Umleitung implementiert. Aufrufe an WriteTTY werden in eine Reihe von Aufrufen zum Schreiben von Zeichen und zur Cursorpositionierung zerlegt. Beachten Sie, daß in OS/2 die TTY-Services als Teil des I/O-Subsystems und nicht im Dateisystem implementiert sind.

Ersetzen von I/O-Subsystem-Funktionen

Da das gesamte OS/2-I/O durch die I/O-Subsysteme geschleust wird, ist dieser Bereich der geeignete Platz, um die

Funktionen »abzufangen«. Die I/O-Subsysteme in OS/2 sind, mit anderen Worten, vollständig durch Applikationsprogramme oder Systemerweiterungen ersetzbar. Die Struktur jedes I/O-Subsystems (siehe *Abb. 3*) enthält eine Router-Komponente und einen Satz Serviceroutinen. Wenn ein Programm (Applikation oder System) von einem der I/O-Subsysteme liest oder eines beschreibt, wird diese Anforderung vom Router verarbeitet und in den meisten Fällen an die vom System gestellte I/O-Serviceroutine weitergegeben. Für jede Subsystem-API wird von OS/2 eine Standard-Serviceroutine angeboten.

Eine Systemerweiterung (entweder Teil der Applikation oder eines anderen Prozesses) kann sich selbst registrieren lassen, um irgendeine oder alle I/O-Subsystem-APIs zu handhaben. Der Router sendet einfach die einzelnen API-Aufrufe an jene System-Serviceroutine oder jene Systemerweiterung, die zur Verarbeitung registriert wurde. Im Zweifelsfall gehen alle I/O-Subsystemaufrufe an die Serviceroutinen, die mit OS/2 angeboten werden. Wenn die Systemerweiterung den API-Aufruf empfängt, kann sie eine der folgenden Aktionen ausführen:

- die I/O-Anforderung verarbeiten;
- die Anforderung prüfen und an die OS/2-Serviceroutine weitergeben (mit oder ohne Modifizierung der Applikationsdaten);
- nichts tun und die Kontrolle an die Applikation zurückgeben (dadurch wird dann die Funktions-Anforderung »verschluckt«).

Abb. 4 illustriert, wie eine Systemerweiterung sich selbst eintragen lassen kann, um einige der I/O-Subsystem-Services zu handhaben. Beachten Sie, daß jeweils nur eine Systemerweiterung registriert werden kann, die alternative Subsystem-Services anbietet. OS/2 erlaubt keine Verkettung von Erweiterungen.

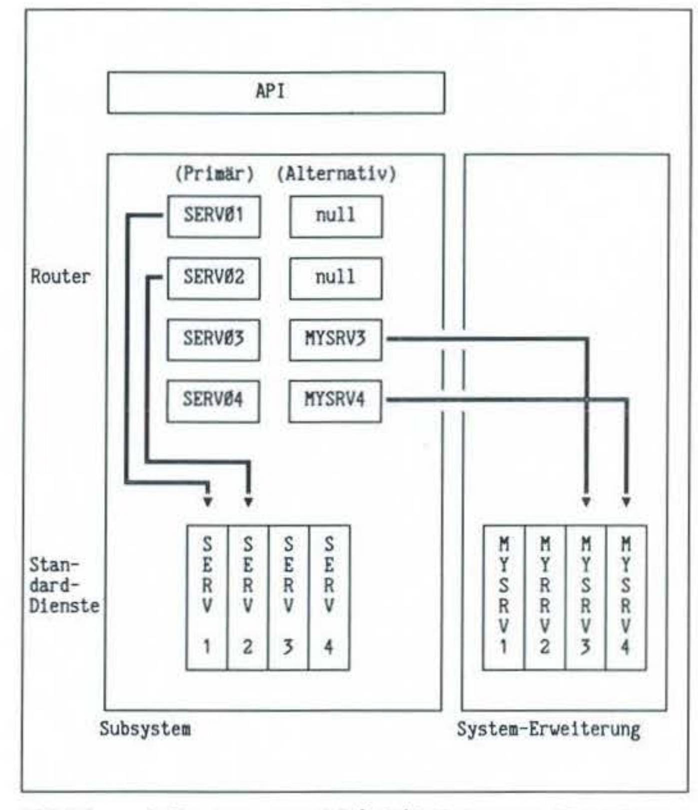


Abbildung 4: *Ersetzen von OS/2-I/O-Subsystem-Funktionen.*

Sessions

OS/2 implementiert mehrfache Anwender-Sessions. Eine Session ist ein Prozeß oder mehrere Prozesse, die zusammen laufen und Tastatur, Videodisplay und die Maus gemeinsam nutzen. In den meisten Fällen entspricht eine Session einem Applikationsprogramm. Der OS/2-Anwender kann bis zu 12 gleichzeitig ablaufende Sessions starten.

Der Begriff Session ist wichtig im Zusammenhang mit I/O-Subsystemen, da er einen logischen Zustand des Subsystems definiert. Obwohl jeweils nur eine Version des Subsystems in das System geladen wird, besitzt doch jede Session einen anderen Kontext. Jede Session verfügt mit anderen Worten über ihren eigenen Satz Datenbereiche, die den Status der Tastatur, des Videodisplays und der Maus definieren. Wechselt der Anwender von Session zu Session, wird der Kontext des Devices wieder zurückgeladen.

Das System unterhält im Fall der Video-Services für jede Session einen logischen Video-Buffer (LVB). Der LVB ist ein Abbild des Bildschirminhalts. OS/2-Programmierer greifen darauf über die Video-Subsystem-Aufrufe zu. Diese Funktionsaufrufe stehen immer zur Verfügung, und eine Applikation kann so den Bildschirm sogar dann aktualisieren, wenn er sich in einer Hintergrund-Session befindet.

Im Fall der Tastatur und der Maus unterhalten die OS/2-Subsysteme für jede Session eine eigene Queue. Wechselt der Anwender die Session, schaltet OS/2 zwischen den Input-Queues um.

Abb. 5 illustriert, wie OS/2 mehrere Sessions unterstützt. Die Ersetzbarkeit von Subsystem-Funktionen (die vorher beschrieben wurden) wird Session-weise durchgeführt. Verschiedene Systemerweiterungen in verschiedenen Sessions können zum Beispiel die I/O-Subsystem-Services übernehmen. Eine Systemerweiterung kann zur Handhabung von I/O-Anforderungen für seine Session nur sich selbst registrieren lassen. Da jedoch die Grund-Services in einem Dynamic-Link-Modul enthalten sind, können sie ersetzt werden (für alle Sessions), indem einfach die DLL-Datei ersetzt wird.

Auswirkungen auf das Family-API

Die meisten OS/2-Subsystem-Services sind im Family-API enthalten. So verbinden denn auch die API-Bindings, die mit Subsystem-I/O-Aufrufen verknüpft sind, die Systemroutinen mit dem Applikationsprogramm. Um denselben Grad an Leistung in der PC-DOS-Umgebung zu bieten, verwenden diese Bindings das System-BIOS nicht.

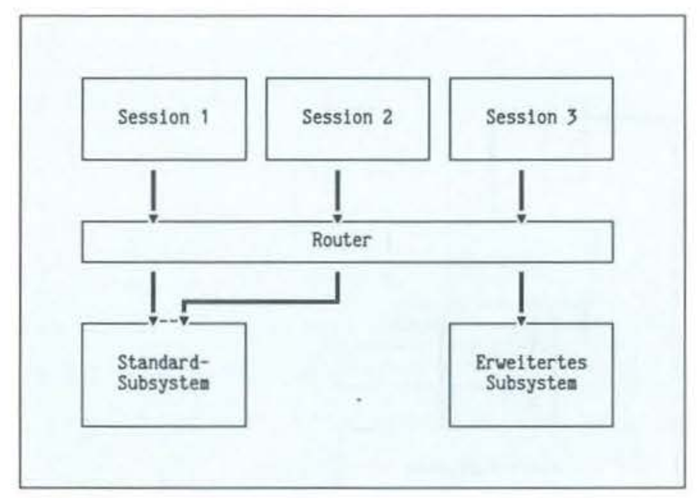


Abbildung 5: *OS/2-Session.*

Das I/O-Subsystem-API ist also ein Satz Standard-I/O-Subroutinen. Wenn eine Family-Applikation unter PC-DOS läuft, führt der in den Bindings enthaltene Code die I/O-Services durch. Läuft dieselbe Applikation unter OS/2 (im Protected-Mode), fallen die Bindings weg und die grundlegenden Subsystem-Services werden eingesetzt.

Video-I/O-Subsystem-Services

Die Video-I/O-Services (die insgesamt als VIO-API bezeichnet werden) sind ein Funktionensatz, äquivalent zu den BIOS-Video(INT10)-Routinen. Die VIO-Services gehen jedoch über die BIOS-Einrichtungen hinaus, da sie Erweiterungen zur Manipulation von Zeichenstrings, Attributen und Zellen in einer funktionelleren Weise enthalten als die BIOS-Services, die sie ersetzen. Eine Zeichenzelle ist die Kombination eines Zeichens und seines Display-Attributs.

Eine erwähnenswerte Ausnahme bilden die Grafikfunktionen (pixelorientiert), die von den VIO-APIs nicht unterstützt werden. Das VIO-Subsystem unterstützt somit das Display nur im Textmodus. Obwohl VIO keine Grafik-Primitive enthält, erlaubt es einer Applikation, Grafik-Display-Modes einzustellen. Wenn eine Applikation das Display auf Grafikmode einstellt, übernimmt sie die Verantwortung für die Handhabung des Displays. Diese Einschränkung ist jedoch nicht so ernst, wie es aussehen mag. Applikationen in der PC-DOS-Umgebung setzen fast nie die BIOS-Services ein, um einzelne Pixel des Bildschirms zu beschreiben. In PC-DOS werden Grafik-Services in der Regel von höheren Grafik-Schnittstellen angeboten, zum Beispiel vom virtuellen Device-Interface (VDI) oder dem Grafik-Device-Interface (GDI).

VIO bietet viele Stufen für den Display-Zugriff. Welche Technik verwendet werden sollte, hängt stark von den spezifischen Eigenschaften des Applikationsprogrammes ab.

Sequenz	Beschreibung
ESC[#;#H	Cursor zu den Koordinaten (ROW;COL) bewegen
ESC[#;#H	Cursor zu den Koordinaten (COL;ROW) bewegen
ESC[#A	Cursor eine oder mehrere Zeilen nach oben bewegen
ESC[#B	Cursor eine oder mehrere Zeilen nach unten bewegen
ESC[#C	Cursor eine oder mehrere Spalten nach rechts bewegen
ESC[#D	Cursor eine oder mehrere Spalten nach links bewegen
ESC[#;#R	Cursorposition zurückgeben
ESC[6n	Device-Status zurückgeben
ESC[s	Cursorposition speichern
ESC[u	Cursorposition wiederherstellen
ESC[2J	Display löschen
ESC[K	Löschen bis Zeilenende
ESC[#;..;#m	Zeichenattribute setzen
ESC[=#h	Displaymode setzen
ESC[=#1	Displaymode zurücksetzen
ESC[#;..;#p	Tastatur-Input erneut zuweisen

Tabelle 1: *ANSI-Escape-Sequenzen*

Virtueller TTY-Output

Mit dem virtuellen TTY-Service des VIO erhalten Applikationen eine höhere Schnittstelle zum Bildschirm. Das VIO-TTY ähnelt in der Funktion dem auf Handle basierenden I/O, das vom Dateisystem angeboten wird. Wenn ein Applikationsprogramm diese Funktion verwendet, ist der Bildschirm-Output nicht umlenkbar. Das Dateisystem verwendet die VIO-TTY-Funktion für die Bildschirmausgabe.

Im TTY-Mode werden Zeichen immer an der aktuellen Cursorposition geschrieben. Nachdem das Zeichen geschrieben wurde, rückt der Cursor um eine Position vor. Wenn das Zeichen in der letzten Position der Zeile geschrieben wurde, springt der Cursor auf den Anfang der nächsten Zeile. Die folgenden Spezialzeichen im Outputstrom werden als Befehle und nicht als Daten behandelt:

- (13) Carriage-Return: Dadurch wird der Output auf der nächsten Zeile fortgesetzt. Ist der Output auf der letzten Zeile des Displays angelangt, rollt der Bildschirm um eine Zeile nach oben.
- (10) Line-Feed: Hat dieselbe Wirkung wie Wagenrücklauf. Folgt sie einem Wagenrücklauf, wird die Zeilenschaltung ignoriert.
- (08) Backspace: Der Cursor bewegt sich um eine Position zurück (ohne zu löschen).
- (09) Tab: Der Cursor bewegt sich zum nächsten Tabulatorstopp. Tab-Stopps gibt es alle 8 Bytes.
- (07) Bell: Ein kurzer Ton erklingt.

Eine Applikation kann die Funktion VIO-TTY jederzeit einsetzen. Ist die Applikation nicht die aktuelle Vordergrund-Session, wird der LVB der Session aktualisiert. Sobald der Anwender die Session wieder in den Vordergrund bringt, werden die Daten korrekt dargestellt.

Die Funktion VIO-TTY implementiert auch die ANSI-Escape-Sequenzen zur Cursorpositionierung, verbesserter Attributkontrolle und Bildschirm-Mode-Kontrolle (siehe Tabelle 1). Der ANSI-Mode kann mit einem VIO-API oder dem ANSI-Utility-Programm an- oder abgeschaltet werden.

Operationen mit Zeichenstrings

Auf der nächstniedrigeren Funktionsstufe enthält das VIO-Subsystem des OS/2 einen kompletten Satz von Zeichen-I/O-Services der unteren Ebene. Diese Services stellen, allerdings mit einigen Erweiterungen, jene Funktionen zur Verfügung, die in der PC-DOS-Umgebung vom BIOS-INT10 angeboten werden. Tabelle 2 vergleicht die BIOS-mit den VIO-Funktionen. Zu den signifikanten Unterschieden gehören: die Fähigkeit, Zeichenstrings (mit oder ohne Attribute) aus mehreren Bytes schreiben zu können; die Fähigkeit, Zeichen, Attribute und Zellen-Strings lesen zu können, sowie vertikales Rollen.

Die Applikationen können die Zeichenfunktionen genauso wie virtuelles TTY jederzeit einsetzen. Ist die Session im Hintergrund, werden Veränderungen in der LVB der Session durchgeführt.

In fast allen Fällen können Sie textorientierte Applikationen ganz mit den VIO-Zeichenfunktionen implementieren. Die Leistung dieser Funktionen ist bedeutend besser als die der entsprechenden BIOS-Services. Die VIO-Routinen sind für den Display-Adapter im System optimiert. Wenn zum Beispiel mit dem CGA-Displayadapter gearbeitet wird, verwenden die VIO-Funktionen sowohl horizontale als auch vertikale Retrace-Synchronisation zur Maximierung des Video-Durchsatzes.

Ein Applikationsprogramm ist vollständig unabhängig von der Hardware, wenn Zeichen-VIO-Funktionen verwendet werden. Das VIO-Subsystem maskiert die Unterschiede zwischen den verschiedenen Display-Adaptern, die von OS/2 unterstützt werden (CGA, EGA, VGA etc.), und optimiert deren Leistung.

Logischer Video-Buffer

Applikationen können auch direkt auf den LVB zugreifen. Das VIO-Subsystem des OS/2 enthält ein API, das einem Programm direkte Adressierbarkeit seines LVB verleiht. Mit dieser Adresse können Programme direkt jeden Teil des logischen Bildschirms aktualisieren. Sobald die Aktualisierung geleistet worden ist, kann der wirkliche Bildschirm-Buffer mit einer einzigen »Show«-Operation aktualisiert werden. Diese Art des Video-I/O ist sehr nützlich bei Applikationen, die große Teile des Bildschirms aktualisieren. Wird der Bildschirm in großen Blöcken aktualisiert, wird die Video-Leistung noch weiter optimiert.

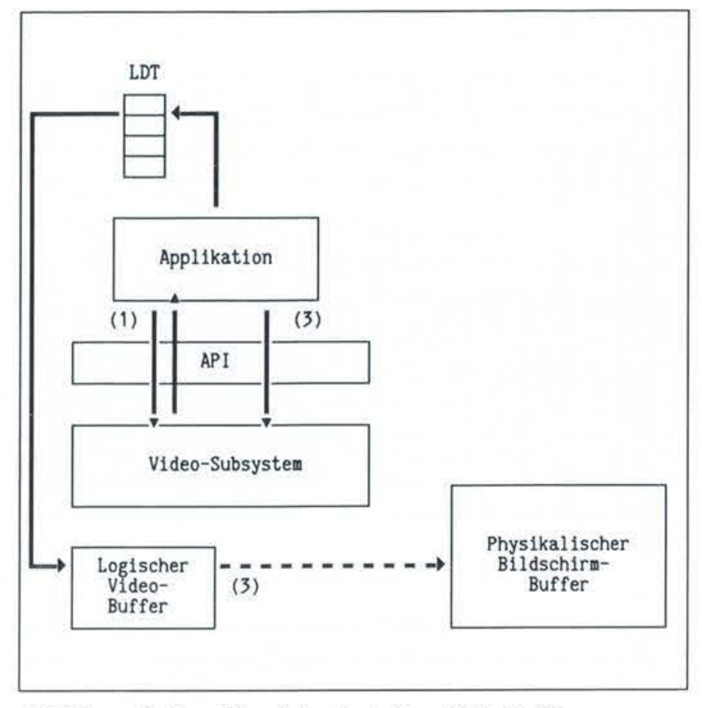


Abbildung 6: *Zugriff auf den logischen Videobuffer.*

Abb. 6 verwendet das folgende Beispiel, um zu zeigen, wie Sie diese Technik beim Aktualisieren des Bildschirms einsetzen können:

1. Die Applikation fordert vom VIO-Subsystem Zugriff auf die LVB an. Ein Selektor wird in der LDT der Applikation erzeugt, und der Selektor wird an das Programm zurückgegeben.
2. Mit diesem Selektor aktualisiert die Applikation die LVB direkt, so, als ob er der wirkliche Video-Buffer wäre. Zu diesem Zeitpunkt werden die Änderungen jedoch nicht auf dem Bildschirm sichtbar.
3. Nach Ausführung aller Aktualisierungen fordert die Applikation, daß die Änderungen auf dem »wirklichen« Bildschirm durchgeführt werden. Das VIO-Subsystem verwendet dann die Daten im LVB, um den physikalischen Bildschirm-Buffer zu ändern.

Der direkte Zugriff auf den LVB ist die Video-I/O-Funktion niedrigster Stufe, die noch Device-Unabhängigkeit bewahrt. Spezielle Device-Eigenschaften bleiben vor der Applikation verborgen. Dieser LVB-Zugriff ist äquivalent zu den INT-10-Unterfunktionen FE und FF, die von TopView in der PC-DOS-Umgebung angeboten werden.

Da die Applikation mit einer logischen Darstellung des Displays arbeitet, können Aktualisierungen der LVB jederzeit (mit der Session im Vorder- oder Hintergrund) durchgeführt werden. Befindet sie sich im Vordergrund, aktualisiert die SHOW-Operation den Bildschirm direkt. Befindet sie sich im Hintergrund, wird die Information einfach gespeichert, bis die Session zum Vordergrund zurückkehrt.

Vorteile für MS Journal ..\etc-Abonennten:

- *Informationen aus erster Hand*
- *Fachberichte über ausgereifte Applikationen*
- *Umfangreiche Testberichte*
- *Veröffentlichung von Leseartikeln*
- *Neue Produkte*
- *Viele Fragen und Antworten zur Technik*

Sichern Sie sich die MS Journal ..\etc auch weiterhin! Der Abo-Preis beträgt seit 1.1.1989 DM 36,– für 1 Jahr (12 Ausgaben) oder DM 64,– für 2 Jahre (24 Ausgaben).

Vorteile für Microsoft System Journal Abonnenten:

- *Experten-Informationen*
- *Fragen und Antworten*
- *Tips & Tricks*
- *Termine & Daten*
- *Information über Bücher & Software*
- *Tools & Routinen*
- *usw., usw., usw.*

Sichern Sie sich die Microsoft System Journal zum Abo-Vorteilspreis von DM 115,– für 1 Jahr (6 Ausgaben), oder von DM 210,– für 2 Jahre (12 Ausgaben).

Vorteile für Microsoft System Journal Abonnenten mit Diskettenbestellung:

- *Listings auf Diskette zum Compilieren*
- *schnelle Übernahme in Programme*
- *lieferbar auf der derzeit am weitest verbreiteten 5¼" Diskette 360 KB für IBM PC und Kompatible*

Sichern Sie sich das Kombinationspaket der gedruckten Ausgabe des Microsoft System Journal und der digitalisierten Daten auf Diskette

MS Journal ..\etc · BESTELLUNG

Ja, ich bestelle die MS ..\etc. ab: ____________

☐ **für mich/uns** ☐ **für umseitigen Empfänger**

Für: ☐ 1 Jahr (12 Ausgaben) zum Vorteilspreis von DM 36,–
☐ 2 Jahre (24 Ausgaben) zum Vorteilspreis von DM 64,–

Postzustellung frei Haus. Vertriebskosten und Mehrwertsteuer sind im Vorteilspreis enthalten.

(Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Dieses Angebot gilt für die Bundesrepublik Deutschland und West-Berlin.
Auslandspreise: Schweiz sfr 40,–, Österreich öS 280,– (für 1 Jahr)
Schweiz sfr 64,–, Österreich öS 500,– (für 2 Jahre)

Microsoft System Journal Disketten – Bestellung

Ja, ich bestelle das Microsoft System Journal + Diskette ab: ________

☐ Für 1 Jahr (6 Ausgaben) zum Vorteilspreis von DM 230,–
☐ für 2 Jahre (12 Ausgaben) zum Vorteilspreis von DM 420,–.

ich bestelle nur die Diskette

☐ Einzeldiskette je Ausgabe	DM 19,80
☐ für 1 Jahr (6 Ausgaben)	DM 118,80
☐ für 2 Jahre (12 Ausgaben)	DM 220,—

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Dieses Angebot gilt nur für die Bundesrepublik Deutschland und West-Berlin.

Auslandspreise: Microsoft System Journal + Diskette:
für 1 Jahr (6 Ausgaben) Schweiz sfr 230,-, Österreich öS 1 800,-
für 2 Jahre (12 Ausgaben) Schweiz sfr 420,-, Österreich öS 3 300,-.

Auslandspreise für Diskettenbestellung

Einzeldiskette	Schweiz sfr 19,80	Österreich öS 150,—
für 1 Jahr (6 Ausgaben)	Schweiz sfr 118,80	Österreich öS 950,—
für 2 Jahre (12 Ausgaben)	Schweiz sfr 220,—	Österreich öS 1 700,—

Ja, ich bestelle das Microsoft System Journal ab: ________

Für:

☐ 1 Jahr (6 Ausgaben) zum Vorteilspreis von DM 115,–
☐ 2 Jahre (12 Ausgaben) zum Vorteilspreis von DM 210,–.

Postzustellung frei Haus. Vertriebskosten und Mehrwertsteuer sind im Vorteilspreis enthalten.

Dieses Angebot gilt nur für die Bundesrepublik Deutschland und West-Berlin.
Auslandspreise:
Schweiz sfr 115,- Österreich öS 850- für 1 Jahr (6 Ausgaben).
Schweiz sfr 210,- Österreich öS 1 600,- für 2 Jahre (12 Ausgaben).

Das Abonnement verlängert sich automatisch um ein weiteres Jahr zu den dann gültigen Preisen, wenn es nicht acht Wochen vor Ablauf gekündigt wird.

Ich bezahle mein Abonnement:

☐ Sofort nach Erhalt der Rechnung.
☐ Durch Bankeinzug.

Bitte die Einzugsermächtigung auf der Rückseite ausfüllen.

MS Journal ..\etc Media-Informationen

❐ JA, bitte schicken Sie mir schnellstmöglich ***MS Journal... \etc*** Media-Unterlagen zu.

❐ JA, ich bin interessiert an Anzeigenschaltungen in ***MS Journal .. \etc.*** Rufen Sie mich bitte unter folgender Telefon-Nummer an :

....

Name

in Firma

Straße

PLZ /Ort

Empfänger der Zeitschrift

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Garantie

Mir ist bekannt, daß ich diese Bestellung innerhalb einer Woche bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung meines Widerrufsschreibens.
Ich bestätige dies durch meine zweite Unterschrift.

Datum/Unterschrift

Bitte freimachen

Antwort

Druck- und Verlagshaus Alois Erdl KG

Postfach

D-8223 Trostberg

Jetzt zum Abo-Vorteilspreis bestellen!



Vorsprung sichern · aus erster Hand informieren.
MS Journal ..\etc
im Jahresabonnement
nur DM 36,–
im 2-Jahresabonnement
nur DM 64,–



Einzugsermächtigung

Hiermit ermächtige ich Sie widerruflich, die von mir zu entrichtenden Zahlungen für bei Ihnen bestellte Artikel bei Fälligkeit zu Lasten meines

Kontos Nr.:

BLZ:

Geldinstitut
durch Lastschrift einzuziehen. Wenn mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Geldinstituts keine Verpflichtung zur Einlösung.

Datum/Unterschrift

Garantie

Mir ist bekannt, daß ich diese Bestellung innerhalb einer Woche bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung meines Widerrufsschreibens.
Ich bestätige dies durch meine zweite Unterschrift.

Datum/Unterschrift

Bitte freimachen

Antwort

System Journal
Leserservice 7311

Postfach 6740

D-8700 Würzburg

Jetzt zum Abo-Vorteilspreis bestellen!



Insider-Informationen für Programmierer, Softwareentwickler, Systemdesigner und erfahrene Anwender von Microsoft-Software.

Microsoft System Journal

im Jahresabonnement (6 Ausgaben) zu DM 115,–
im 2-Jahresabonnement (12 Ausgaben) zu DM 210,–

auch mit Diskette

Listings auf Diskette zum Compilieren oder zur schnellen Übernahme in Programme. Lieferbar auf 5¼"Disketten 360 KB für IBM PC und Kompatible

Microsoft System Journal + Diskette

im Jahresabonnement (6 Ausgaben) zu DM 230,–
im 2-Jahresabonnement (12 Ausgaben) zu DM 420,–

Absender:

Bitte mit 60 Pfg. freimachen

Antwort

MARKETING PROJEKT 2000 GmbH
Frau Janz

Gottfried-Böhm-Ring 59

D-8000 München 70





BIOS	VIO-API	Beschreibung
00	VioSetMode	Displaymodus setzen
01	VioSetCurType	Cursor-Typ setzen
	VioGetCurType	Cursor-Typ ausgeben
02	VioSetCurPos	Cursor-Position setzen
03	VioGetCurPos	Cursor-Position lesen
04		Lichtgriffel-Position lesen (Lichtgriffel wird von VIO nicht unterstützt)
05		Aktive Seite wählen (Mehrere Seiten werden von VIO nicht unterstützt)
06	VioScrollUp	Seite aufwärtsrollen
07	VioScrollDn	Seite abwärtsrollen
	VioScrollLf	Seite nach links rollen
	VioScrollRt	Seite nach rechts rollen
08	VioReadCellStr	Zelle lesen (VIO liest String)
	VioReadCharStr	Zeichen lesen (VIO liest String)
09	VioWrtCharStr	Zelle an Cursor-Position schreiben
	VioWrtNCell	Zelle wiederholen
10	VioWrtCellStr	Zeichen an Cursor-Position schreiben
	VioWrtNChar	Zeichen wiederholen
	VioWrtNAttr	Attribut wiederholen
11	VioSetPal	Farbpalette setzen
12		Grafikpunkt schreiben (Grafik wird von VIO nicht unterstützt)
13		Grafikpunkt lesen (Grafik wird von VIO nicht unterstützt)
14	VioWrtTTY	TTY-String schreiben (ANSI in VIO enthalten)
15	VIOGetMode	Video-Status ausgeben
1900	VioWrtCharStrAtt	Zeichen-String schreiben (Mit demselben Attribut)
1901	VioWrtCharStrAtt	Zeichen-String + Cursor schreiben (Mit demselben Attribut)
	VioWrtCharStr	Zeichen-String schreiben (Nur Zeichen, keine Attribute)
1902	VioWrtCellStr	Zell-String schreiben
1903	VioWrtCellStr	Zell-String + Cursor schreiben

Tabelle 2: *BIOS-INT10- und VIO-Text-Funktionen*

Abhängig von den Eigenschaften der Applikation sollte entschieden werden, ob statt der VIO-Zeichenfunktionen der LVB direkt verwendet werden sollte. Eine Applikation, die kleine Teile des Bildschirms aktualisiert, eignet sich am besten für die Zeichen-I/O-Funktionen. Umgekehrt lassen sich Aktualisierungen großer Blöcke am besten mit LVB-Manipulationen durchführen. VIO-Zeichenfunktionen und LVB-Aktualisierungen schließen sich nicht gegenseitig aus: Eine Applikation kann in verschiedenen Situationen wechselweise beide einsetzen.

Zugriff auf das physikalische Display

Auf der untersten Stufe erteilt das VIO-Subsystem des OS/2 direkten Zugriff auf den physikalischen Display-Buffer. Da OS/2 ein Multitasking-System ist, müssen spezielle Vorkehrungen getroffen werden, um sicherzustellen, daß die Bildschirm-Ressource vernünftig geteilt wird. Das VIO-Subsystem enthält mehrere Mechanismen, die eingesetzt werden müssen, wenn ein Applikationsprogramm direkt Bildschirm-I/O durchführt.

Beachten Sie, daß es in den meisten Fällen weder notwendig noch erwünscht ist, ein Applikationsprogramm zu schreiben, das direkt Bildschirm-I/O durchführt. Die Leistung der Zeichen-VIO-Funktionen des OS/2 ist adäquat für fast jede Applikation. Reicht die Leistung nicht, kann Bildschirm-I/O durch direkte Anwendung der VIO-LVB optimiert werden. Der direkte Zugriff auf das Display sollte aber nur als letzte Rettung erwogen werden. In gewissen Fällen, wie zum Beispiel bei Grafik-Applikationen, ist physikalischer Zugriff jedoch obligatorisch.

Wenn eine Applikation das Display direkt beschreibt, muß es auch die physikalischen Eigenschaften des Bildschirms kennen. Physikalischer Bildschirm-I/O wird deshalb als Device-abhängige Applikationsfunktion bezeichnet.

Gemeinsame Nutzung des Display-Buffers

Ähnlich wie Sie den LVB anfordern, setzen Sie ein spezielles OS/2-API ein, um direkte Adressierbarkeit des physikalischen Bildschirm-Buffers zu verlangen. Das VIO-Subsystem erzeugt einen LDT-Deskriptor, der den Display-Buffer abbildet und Ihrer Applikation einen Selektor für den Zugriff auf das Speichersegment gibt. Da im System jedoch nur ein wirklicher Bildschirm-Buffer vorhanden ist, kann der Selektor nur eingesetzt werden, während die Applikation im Vordergrund ist. Schaltet der Anwender die Session in den Hintergrund, wird der LDT-Deskriptor ungültig gemacht. Versucht das Programm, ihn in diesem Zustand zu benutzen, erzeugt die CPU einen General-Protection-Fehler, und das Applikationsprogramm wird von OS/2 terminiert. *Abb. 7* illustriert, wie mehrere Sessions den Display-Buffer gemeinsam nutzen.

Synchronisation von Bildschirmzugriffen

Es ist sehr wichtig, daß ein Programm nur dann auf das physikalische Display-Device zugreift, wenn es in der Vordergrund-Session ist. Zur Entscheidungshilfe, wann es sicher ist, den Bildschirm zu beschreiben, bietet OS/2 einen Display-Synchronisations-Service an.

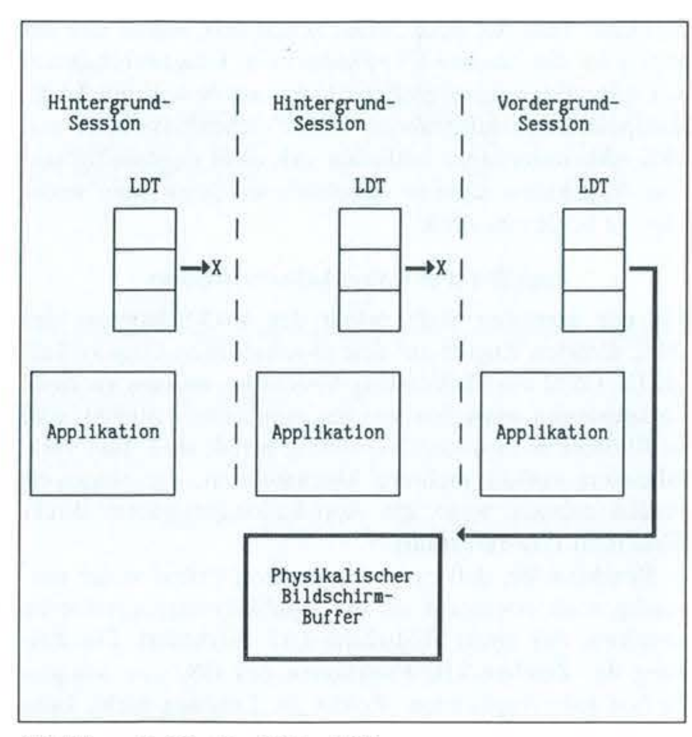


Abbildung 7: *Direktes Video-I/O.*

VIO API	Beschreibung
VioRegister	Alternative VIO-Routine(n) definieren
VioDeRegister	Alternative VIO-Routine(n) löschen
VioGetBuf	Adresse des logischen Video-Buffers (LVB) holen
VioShowBuf	Physikalisches Display mit dem LVB auffrischen
VioGetPhysBuf	Adresse des physikalischen Video-Buffers holen
VioScrLock	Physikalischen Bildschirm für I/O sperren
VioScrUnlock	Sperre des physikalischen Bildschirms lösen
VioSavRedrawWait	Thread für Save/Restore ausweisen
VioSavRedrawUndo	Letzten SavRedrawWait-Aufruf rückgängig machen
VioPopUp	Popup-Fenster einblenden
VioEndPopUp	Popup-Fenster entfernen
VioGetANSI	ANSI-Status holen
VioSetANSI	ANSI-Status setzen (ein oder aus)
VioPrtSc	Bildschirm drucken (äquivalent zu BIOS-INT5)
VioPrtScToggle	PrtSc-Zustand abfragen
VioGetConfig	Video-Konfiguration abfragen
VioGetFont	Aktuellen Zeichensatz abfragen (EGA, VGA)
VioSetFont	Video-Zeichensatz definieren
VioGetCP	Aktuelle Code-Page abfragen
VioSetCP	Code-Page definieren

Tabelle 3: *VIO-Funktionen.*

Jedesmal, wenn ein Programm physikalischen Bildschirm-I/O ausführen möchte, fordert das Programm mit dem Aufruf VioScrLock Exklusivverwendung des Displays an. Befindet sich die Session im Hintergrund, wird das Programm solange unterbrochen, bis der Anwender wieder in den Vordergrund umschaltet, und zu dem Zeitpunkt wird die Bildschirm-LOCK-Funktion erfolgreich realisiert. Befindet sich das Display im LOCK-Zustand, erlaubt OS/2 dem Anwender nicht, in eine andere Bildschirm-Session umzuschalten. Mit der Funktion VioScrUnlock zeigt das Programm an, wenn es mit dem Beschreiben des Bildschirms fertig ist. Der Bildschirm kann also nur dann aktualisiert werden, wenn er sich im LOCK-Zustand befindet.

Die Applikation ist zuständig für das Aufteilen großer Bildschirm-Aktualisierungen in kleinere Elemente, um sicherzustellen, daß der Session-Manager nicht für längere Zeitspannen ausgesperrt bleibt.

Speichern und Wiederherstellen des Video-Status

Für alle Applikationsprogramme, die in Textmodi laufen, speichert OS/2 den Bildschirm-Kontext automatisch und lädt ihn auch wieder zurück. Wenn jedoch das Display-Device in einen Grafikmodus eingestellt wird, wird das Applikationsprogramm für das Abspeichern des Display-Kontextes verantwortlich.

Eine Spezialfunktion, VioSavRedrawWait, führt diese Aufgabe durch. Das Applikationsprogramm erzeugt einen speziellen Ausführungsthread, dessen einzige Aufgabe darin besteht, den Display-Kontext zu speichern und wieder zurück zu laden. Dieser Thread (er wird SAVEREDRAW-Thread genannt) gibt sich dem VIO-Subsystem durch die Ausgabe des Funktionsaufrufes SaveRedraw zu erkennen. Sobald der Aufruf ausgegeben wurde, unterbricht das System den Thread. Der SAVEDRAW-Thread wird unmittelbar bevor der Status der Session von Vordergrund auf Hintergrund oder umgekehrt geändert wird, »aufgeweckt«. Ist der Thread »wach« geworden, prüft er einen Indikator, der dem Thread mitteilt, ob der Status des Bildschirms gespeichert oder wiederhergestellt werden soll. Der Thread führt dann die Save- oder Restore-Funktion durch und gibt den VioSavRedrawWait-Aufruf erneut aus.

Der SAVEDRAW-Thread ist also eine von der Applikation aufgebotene Funktion, die den Zustand des Bildschirms versteht und vom VIO-Subsystem aufgerufen werden kann, um den Inhalt des Bildschirms für die Applikation zu speichern oder wiederherzustellen.

Video-Popups

Mitunter müssen Hintergrund-Programme dringend mit dem Anwender kommunizieren. Es kann zum Beispiel sein,

daß ein Drucker-Spooler den Zustand »Kein Papier mehr« anzeigen möchte, oder daß ein Alarm-Programm eine Nachricht präsentieren muß. Diese Nachrichten sollten offensichtlich nicht warten, bis der Anwender zur richtigen Session wechselt.

Für diese Situationen können Hintergrund-Programme eine spezielle VIO-Funktion VioPopUp aufrufen, um eine Nachricht auf dem Bildschirm darzustellen. Ein OS/2-Popup ist jedoch keine Mehrzweck-Funktion, da jeweils nur ein Popup aktiv sein kann. Verlangen mehrere Programme ein Popup, wird eines bedient und die anderen blockiert. Applikationen sollten die Anwendung dieser Einrichtung auf kritische Situationen beschränken, die Anwenderaktionen erfordern.

Kontrolle des Videomodus

Der Funktionsaufruf VioSetMode setzt die Display-Video-Modes. Das VIO-Subsystem unterstützt zwar nur Textoperationen, Applikationen können jedoch den Aufruf dazu verwenden, alle vom Displayadapter unterstützten Video-Modes zu setzen. Dies schließt 40- und 80-Spalten-Text (mit oder ohne Farbe-Burst), einfarbigen Text und alle Grafikmodi ein.

Wenn zwei Displays an die Maschine angeschlossen sind, wechselt das VIO-Subsystem zu dem Device, das den gewählten Mode am besten approximiert. Wenn zum Beispiel ein Monochrom- und ein CGA-Adapter installiert sind, aktiviert zum Setzen des MONO-Modus das monochrome Display und zum Setzen von 80-Zeilen-Farbtext das CGA-Display.

Die Funktion VioGetMode wird eingesetzt, um den aktuellen Display-Mode zu erfragen. *Tabelle 3* faßt die übrigen VIO-Funktionen zusammen.

Tastatur-I/O-Subsystem-Services

Die Tastatur-I/O-Services, die als KBD-API bezeichnet werden, bilden einen Satz Funktionen, der äquivalent zu den BIOS-Tastatur(INT16)-Routinen ist. Zusätzlich umfaßt KBD eine Reihe von Erweiterungen zum Empfang von Zeichenstrings und zur Bestimmung der Umschalt-Zustände. Die KBD-Funktionen sind wie im Fall der VIO-Services in zwei logischen Stufen implementiert.

Höhere KBD-Input-Services

Wie im Fall des VIO-Subsystems enthält auch das KBD-Subsystem eine höhere Schnittstelle zur Tastatur. Die Funktion KbdStringIn ist das Input-Analogon zu den VIO-TTY-Services. Diese Funktion ist ähnlich dem vom Dateisystem angebotenen, auf Handle basierenden I/O. Wenn ein Applikationsprogramm diese Funktion anwendet, ist der Input jedoch nicht umleitbar. Das Dateisystem verwendet die Funktion KbdStringIn, um Zeichen von der Tastatur zu lesen.

BIOS	KBD-API	Beschreibung
00	KbdCharIn	Nächstes ASCII-Zeichen lesen (Warte-Indikator eingeschaltet)
01	KbdCharIn	Testen, ob Zeichen anliegt (Warte-Indikator ausgeschaltet)
	KbdPeek	
02	KbdGetStatus	Shift-Zustand liefern
	KbdCharIn	
	KbdPeek	
-	KbdPeek	Zeichen anschauen (Verbleibt in der Input-Queue)
-	KbdFlushBuffer	Input-Buffer löschen
-	KbdSetStatus	Tastatur-Status setzen
-	KbdStringIn	Zeichen-String lesen (ANSI-Sequenzen werden unterstützt)
Andere Funktionen		
	KbdRegister	Alternative KBD-Routine(n) definieren
	KbdDeRegister	Alternative KBD-Routine(n) löschen

Tabelle 4: *BIOS-INT16 und OS/2-KBD-Funktionen.*

MOU API	Beschreibung
MouRegister	Alternative Routine(n) definieren
MouDeRegister	Alternative Routine(n) löschen
MouOpen	Subsystem für die Session öffnen
MouClose	Subsystem schließen
MouFlushQue	Ereignis-Queue löschen
MouGetDevStatus	Status-Flags des Zeiger-Device abfragen
MouGetEventMask	Ereignis-Maske abfragen
MouGetHotKey	System-Hot-Key abfragen
MouGetNumButtons	Anzahl der Tasten abfragen
MouGetNumMickeys	Anzahl »Mickeys« pro Zentimeter abfragen
MouGetNumQueEl	Status der Ereignis-Queue des Zeiger-Devices abfragen
MouGetPtrPos	Aktuelle Koordinaten des Zeiger-Devices abfragen
MouGetPtrShape	Form des Zeigers abfragen
MouGetScaleFact	Skalierungsfaktoren abfragen
MouReadEventQue	Ereignis-Queue des Zeiger-Device lesen
MouSetEventMask	Ereignismaske setzen
MouSetHotKey	System-Hot-Key setzen
MouSetPtrPos	Aktuelle Zeigerposition setzen
MouSetPtrShape	Form des Zeigers setzen
MouSetScaleFact	Skalierungsfaktoren setzen

Tabelle 5: *OS/2-Maus-Subsystem-Funktionen*

Eine Applikation kann den Aufruf KbdStringIn jederzeit einsetzen. Ist das Programm gerade nicht im Vordergrund, wird der Thread blockiert und wartet solange, bis die Session in den Vordergrund kommt.

Die Funktion KbdStringIn implementiert auch die Standard-ANSI-Escape-Sequenzen für Input-Filterung (siehe *Tabelle 1*). Der ANSI-Mode wird mit einem VIO-API oder dem ANSI-Utility-Programm gesetzt. Im Zweifelsfall ist ANSI-Modus eingeschaltet. Beachten Sie, daß die ANSI-Sequenzen nur dann interpretiert werden, wenn der Input mit dieser Funktion (oder mit Dateisystem-Handle-Input) gelesen wird.

Niedrigere KBD-Input-Services

Das KBD-Subsystem unterstützt auch niedrigere Zeichen-I/O-Funktionen. Programme können diese Aufrufe einsetzen, um die Tastatur-Scan-Codes und Shift-Zustände zu prüfen, um den Input-Buffer zu löschen und um in die Input-Queue zu schauen. Diese Funktionen sind analog der BIOS-INT16-Schnittstelle in der PC-DOS-Umgebung. *Tabelle 4* erläutert die Unterschiede zwischen den BIOS-Services und den niedrigen KBD-Funktionen.

Maus-I/O-Subsystem-Services

Das OS/2-Maus-Subsystem hat im Gegensatz zu Tastatur und Video kein direktes BIOS-Äquivalent. Dieses Subsystem versieht Applikationen mit einer kompletten Bandbreite an Services, die zur Verwaltung des Zeiger-Devices benötigt werden. Maus-I/O ist weder umleitbar, noch enthält es eine ANSI-Unterstützung. *Tabelle 5* faßt die vom MOU-Subsystem angebotenen Funktionen zusammen.

Zusammenfassung

Die OS/2-Subsysteme erweitern das PC-DOS-I/O-Modell ganz erheblich. Diese Komponenten bieten alle Hochleistungs-Primitiva, die zur Erstellung der OS/2-Anwender-Schnittstelle erforderlich sind. Wichtige Punkte, die man im Gedächtnis haben sollte, sind:

- das Subsystem-Modell;
- Beziehung zwischen OS/2-Subsystem und PC-DOS-I/O-Modell;
- I/O-Subsystem-Struktur und -Komponenten;
- Ersetzung von Subsystem-Funktionen;
- OS/2-Sessions;
- Video(VIO)-, Tastatur(KBD)- und Maus(MOU)-I/O-Funktionen.

Ed Iacobucci

Dieser Artikel ist ein Auszug aus dem Buch »Das OS/2-Buch« von Ed Iacobucci, erschienen im McGraw-Hill Verlag, mit dessen freundlicher Genehmigung der Abdruck erfolgt.

Computerbegriffe, Übersetzungshilfen, Marktübersichten:

Nachschlagewerke über Computer und Software

Die Flut nimmt kein Ende: In den letzten Wochen sind wieder so viele interessante Bücher in der Redaktion eingetroffen, die wir leider nicht alle lesen, geschweige denn ausführlich zu besprechen können. Einige sollen jedoch zumindest mit kurzer Inhaltsangabe und bibliographischen Informationen vorgestellt werden. Mittlerweile haben sich eine ganze Reihe Nachschlagewerke angesammelt, denen wir diese Seite widmen wollen. Bei diesen Büchern geht es im wesentlichen um die Erläuterung von Computerbegriffen, um Übersetzungshilfen und um eine Marktübersicht der in Deutschland angebotenen PC-Software.

Wörterbücher

Die beiden vorgestellten Wörterbücher dienen hauptsächlich zur Übersetzung englischer Fachausdrücke. Aus der DDR kommen die beiden geschickt aufgeteilten Bände des »Fachwörterbuch Informatik«. Im ersten Band findet man neben den englischen Fachausdrücken die deutschen, französischen und russischen Übersetzungen. Will man nun aber umgekehrt übersetzen, nimmt man den zweiten, den Register-Band und bekommt dort unter dem gewünschten Begriff eine Kennung, bestehend aus einem Buchstaben und einer vierstelligen Zahl, mitgeteilt. Ausgerüstet mit dieser Kennung, kann man nun im ersten Band die englische, französische und russische Übersetzung in einer Zeile leicht auffinden.

Bei der Auswahl der etwa 25.000 Fachwörter wurden die allgemeinen Grundlagen der Informatik und der Programmierung, die Softwareentwicklung, die Hardwarekomponenten, die Systemzuverlässigkeit und Anwendungen der Informatik in der Industrie berücksichtigt. Nicht berücksichtigt wurden allerdings Begriffe wie Errorlevel, Window oder gar Thread, die bekanntlich aus MS-DOS, Windows bzw. MS OS/2 stammen.

Prof. Dr.-Ing. habil Erich Bürger (Hrsg.): »Fachwörterbuch Informatik«, Heidelberg, Hüthig Verlag; 1989; 2 Bände, 903 Seiten; ISBN 3-7785-1586-1; DM 138,-.

Mehr hardwareorientiert zeigt sich das sogenannte »Große IWT-Wörterbuch«, das sich an technische Autoren, Übersetzer, Informatiker und an Technik interessierte Laien richtet. Es enthält über 23.000 Begriffe der Elektronik und Mikroelektronik, ist klassisch gegliedert in einen englisch-deutschen sowie deutsch-englischen Teil. Die Zielsetzung dieses Werkes war es, den neuesten Stand der Terminologie am Sprachgebrauch der Praxis darzustellen - was im großen und ganzen auch gelungen ist. Allerdings werden z.B. neurale Netze (neuron network simulation) etwas einfallslos als Neuronen-Netzwerk-Simulation bezeichnet.

Dorothée Jauß, Cipriano Villani: »Großes IWT-Wörterbuch der Elektronik und Mikroelektronik«, Vaterstetten, IWT Verlag, 1989; 661 Seiten; ISBN 3-88322-218-6; DM 78,-.

Computerbegriffe kurz erläutert

Einer etwas anderen Spezies gehören die beiden nun folgenden Bücher an. Sie erläutern die wichtigsten Computerbegriffe kurz und prägnant. Das nunmehr in der zehnten Auflage vorliegende »Mikrocomputerlexikon« wurde überarbeitet und um aktuelle Begriffe erweitert. Das macht sich z.B. dadurch bemerkbar, das solche Fachausdrücke wie Semaphore oder Thread zu finden sind. Behandelt wird alles von den Grundbegriffen bis zu den neueren Anwendungen wie Desktop Publishing und C-Techniken. Außerdem sind im Anhang noch ein englisch-deutsches Wörterbuch und diverse nützliche Tabellen zu finden.

Helmut Kraus: »Mikrocomputerlexikon«, Düsseldorf: Sybex-Verlag, 1989; 315 Seiten; ISBN 3-88745-518-5; DM 14,80.

Etwas ausführlicher werden die Begriffe im »Software-Lexikon« erläutert, das sich im Gegensatz zum oben vorgestellten Titel jedoch ausschließlich mit der Software und im engeren Sinne mit der Programmierung beschäftigt. In über 1.000 Stichwörtern werden nicht nur die wichtigen Programmiersprachen und die strukturierte Programmierung beschrieben, auch andere Themen wie die Qualitätsanforderungen an Software und die wichtigsten Normen kommen nicht zu kurz.

Prof. Klaus W. Jamin: »Das Software-Lexikon«, Stuttgart: Taylorix-Fachverlag, 1988; 348 Seiten; ISBN 3-7992-0439-3; DM 56,-.

Software-Marktübersicht

Bevor Sie das Rad wieder neu erfinden oder sich vielleicht entschließen, ein »runderes« zu konstruieren, können Sie im »PC Softwarebuch '89« nachschlagen. Hier finden Sie anhand des praktischen Registers vielleicht schon ein Programm mit den Leistungsmerkmalen, die Sie brauchen.

Das Spektrum der vorgestellten Programme reicht von branchenbezogener und neutraler Software bis hin zu wissenschaftlich-technischen Programmen, Systemsoftware sowie Lehr und Trainingsprogrammen. Die Programmbeschreibungen werden ergänzt durch Angaben wie z.B. Hardwarevoraussetzungen, Programmiersprache, Benutzerführung, Service, Zahl der Installationen, Bezugsquelle und Preis. Die Register Programmart, Hardware, Anbieter und Programmnamen sorgen für einen schnellen Zugriff auf die gewünschten Informationen.

Das Softwarebuch erscheint jährlich neu mit den aktuellen Informationen, die auf den Meldungen der Softwarehäuser basieren.

»PC Softwarebuch '89«, Haar: Markt & Technik Verlag, 1989; 814 Seiten; ISBN 3-89090-751-2; DM 39,-.

Mitteilungen .... Software .... Mitteilungen

Der 1. Microsoft System Workshop

Zum »1. Microsoft System Workshop für Windows- und OS/2-Entwickler« lud die Microsoft am 18. April nach München. Rund 160 Entwickler aus der Bundesrepublik, Schweiz und Österreich kamen und tauschten in mehreren Arbeitsgruppen ihre Programmiererfahrungen aus. In Kleingruppen konnten die Systemköche einmal ohne interessierte Presse und ohne Öffentlichkeit ungestört Probleme beim objektorientierten Programmieren, Test und Debugging-Verfahren, Optimierungsverfahren, Interprozeßkommunikation und LAN Manager Programmierung erörtern. K.P. Welch, Präsident der amerikanischen Eikon Systems und P. Weger, Leiter des Supports beim LAN-Spezialisten 3COM referierten neben Experten der deutschen Microsoft-Niederlassung. Unter den Teilnehmern waren viele bekannte Namen. Kaum eines der bedeutenden Systemhäuser fehlte auf der Liste. Große Häuser wie Siemens, die sich schon längere Zeit intensiv mit der Entwicklung von Windows- und OS/2-Anwendung befassen, rückten teilweise gleich mit halben Dutzendschaften an. Einig war man sich, daß dies nicht der letzte Workshop dieser Art gewesen sein wird. Einige Teilnehmer wollen sich regelmäßig in informellen Arbeitsgruppen treffen.

Word 5.0 für MS-DOS und MS OS/2 angekündigt

Nach vielen Gerüchten ist es nun endlich soweit: Die englische Version 5.0 des Textverarbeitungsprogramms Microsoft Word für MS-DOS und MS OS/2 wurde fertiggestellt. Die deutsche Fassung wird voraussichtlich im September dieses Jahres in den Handel kommen.

Die Version 5.0 enthält gegenüber ihrem Vorgänger zahlreiche neue Funktionen, u.a.:

- Unterstützung des Protected Mode von MS OS/2;
- Preview-Funktion;
- Spaltenverarbeitung auf dem Bildschirm;
- erweiterte Dateiverwaltungsfunktionen.

Da die Version 5.0 über die gleiche Menüführung verfügt wie Microsoft Word 4.0, ist ein Umstieg problemlos möglich.

Trotz des erhöhten Leistungsumfangs wird der Preis von Microsoft Word 5.0 gleichbleiben und nach wie vor 1.490 DM zzgl. MwSt (unverb. Preisempf.) betragen. Die Schulversion wird für 410 DM zzgl. MwSt angeboten. Der Update-Preis von Microsoft Word 4.0 auf die Version 5.0 liegt bei 399 DM zzgl. MwSt. Für Anwender allerdings, die nach dem 1. Mai 1989 die deutsche Version 4.0 erworben haben bzw. erwerben, wird ein Update auf Microsoft Word 5.0 (deutsch) zum Sonderpreis von 179 DM zzgl. MwSt angeboten, der bis zum 31. Dezember 1989 gewährt wird.

Interessant ist eine Aktion, bei der Microsoft-Händler ab Juni 1989 eine begrenzte Stückzahl sogenannter »Microsoft Word 4.0 Sommerpakete« anbieten. Zum Preis von 1.490 DM zzgl. MwSt »steckt« in diesem Paket neben Word 4.0 ein Gutschein, mit dem man zusätzlich bei Microsoft wahlweise ein Booklet mit Druckformatvorlagen (inklusive Diskette) oder ein Bitstream/Word-Installationskit mit den Softfonts Swiss und Dutch für Laserdrucker und PostScript-Drucker erwerben kann.

Microsoft Multiplan 4.0 ab sofort im Handel

Ab sofort bietet Microsoft das Tabellenkalkulationsprogramm Multiplan 4.0 für die beiden Betriebssysteme MS-DOS und MS OS/2 an. Die neue Version (Preis: 864 DM zzgl. MwSt.) zeichnet sich gegenüber der Vorversion durch eine Reihe weiterer Verbesserungen aus und stellt die erste große Standardanwendung von Microsoft für das neue Betriebssystem MS OS/2 dar.

Anwender von Microsoft Multiplan 3.0 können für einen Aufpreis von 120 DM (inkl. MwSt.) bis einschließlich 31.07.89 auf die neue 4.0 Version überwechseln. Danach gilt der Preis der jeweils aktuellen Preisliste. Die Update-Version ist beim Fachhandel oder direkt bei Microsoft gegen Einsendung der Originaldisketten und des Handbuchs erhältlich.

Eigenschaften von Microsoft Multiplan 4.0

Datenbank

Um Listen besser auswerten zu können und nach bestimmten Informationen zu suchen, lassen sich mit Microsoft Multiplan 4.0 Tabellen oder Teile dieser Listen als Datenbank organisieren. Mit Befehlen können Daten nach bestimmten Kriterien gesucht und auf Wunsch in eine neue Datei extrahiert werden. Darüber hinaus bietet Microsoft Multiplan 4.0 statistische Funktionen für die Arbeit mit der Datenbank.

MS OS/2-Unterstützung

Multiplan 4.0 ist jetzt auch unter dem Betriebssystem MS OS/2 (im Protected Mode) ablauffähig. Multiplan kann also gleichzeitig mit anderen Anwendungsprogrammen ausgeführt oder mehrmals gestartet werden, um die verschiedensten Aufgaben zu bearbeiten. Diese Unterstützung der Multitasking-Eigenschaften führt zur optimalen Rechnerausnutzung und Zeitersparnis.

Verbesserte Feldformatierung

Microsoft Multiplan 4.0 ermöglicht mit den Formatierungen »Fett«, »Kursiv« und »Unterstrichen« eine Hervorhebung von Zellen oder Bereichen von Tabellen. Damit können Modelle besser strukturiert und übersichtlicher gestaltet werden. Die Farbe von Daten in Feldern und Zahlenformate können vom Wert der Zelle abhängig gemacht und wichtige Ergebnisse besonders hervorgehoben werden.

Funktionsauswahl

Multiplan 4.0 weist mehr als 30 neue Funktionen auf, darunter Datenbankfunktionen, finanzmathematische, mathe-

Mitteilungen .... Software .... Mitteilungen

matische und statistische Funktionen, sowie Sonder- und Textfunktionen, die die Formelerstellung erleichtern. Sucht man den Namen einer bestimmten Funktion, kann man am Bildschirm in einer Liste aller Funktionen nachsehen und diesen in die Formel einfügen, ohne im Handbuch nachzuschlagen.

Makros

Multiplan 4.0 weist mehrere neue Makrobefehle auf, die das Aus- und Einschalten der Menü- und Bildschirmaktualisierung ermöglichen und das Bearbeiten von Eingaben in Befehlsfeldern erleichtern.

Ändern der Feldbreite

Multiplan 4.0 ermöglicht die Verwendung von Richtungstasten zum Ändern der Feldbreiten im Menüpunkt »Format Breite der Spalten« und »Format Standard Breite der Spalten«. Mit jedem Tastendruck ändert sich die Spaltenbreite am Bildschirm, so daß sie dort direkt überprüft werden kann.

Kompatibilität mit alten Menüs

Die Befehlsmenüs wurden in Multiplan 4.0 zum Teil geändert. Um trotzdem die unter Microsoft Multiplan 3.0 erstellten Makros einsetzen zu können, läßt sich auf das Menü der Version 3.0 umschalten.

Druckertreiber

Mit Multiplan 4.0 können Tabellen ohne Eingabe von Druckersteuerzeichen jetzt einfach gestaltet und formatiert werden. Man wählt den gewünschten Druckertreiber (wie bei Microsoft Word), um eine Tabelle mit den Schriftarten und -größen zu versehen, die im Bildschirmmenü angeboten werden.

Einstellen der Druckoptionen

Die Ränder der zu druckenden Tabelle lassen sich in Anzahl der Zeichen (Ze), Zoll (In) oder Zentimeter (Cm) spezifizieren.

Ordnen

Es kann nach mehreren Zeilen oder Spalten geordnet werden. Jede spezifizierte Zeile bzw. Spalte wird als ein separater Sortierschlüssel behandelt, wobei die erste Zeile bzw. Spalte in der Liste die wichtigste ist. Mit diesem Befehl kann nach bis zu 20 Schlüsseln gleichzeitig sortiert werden.

Änderungen zum Befehl »Übertragen Laden«

Der Befehl »Übertragen Laden« ist jetzt kontextabhängig: d.h. Multiplan 4.0 lädt Tabellen und Dateien, die im Multiplan-, SYLK- oder ASCII-Format gespeichert sind, automatisch in dem Format, das im Befehl »Übertragen Optionen« zuvor festgelegt wurde.

Speichern eines Tabellenbereichs

Teilbereiche einer Tabelle können durch Spezifizieren des Speicherbereichs im Befehl »Übertragen Optionen« gespeichert werden, um z.B. eine sehr große Datei in übersichtlichere Tabellen zu splitten.

Darstellung des Befehlsbereichs

Die Farbe von Befehlen kann zur leichteren Unterscheidung von Befehlsnamen und Optionen geändert werden.

Neue Tools für OS/2 Presentation Manager

Mit Beginn der Verfügbarkeit von OS/2 Version 1.1 (Presentation Manager) bei IBM und führenden OEMs (NCR, Bull, Compaq u.a.) hat sich die Situation bei den Entwicklungssystemen, die bei Microsoft für OS/2 erhältlich sind, gravierend geändert. Das Software Development Kit (SDK) und das Network Development Kit (NDK) sind nicht mehr lieferbar. SDK-Besitzer erhalten selbstverständlich noch das letzte Update auf die endgültige OS/2-Version 1.1 mit dem aktuellen LAN-Manager-Release. Alle im SDK enthaltenen Tools, Programmiersprachen und die Dokumentation können nunmehr getrennt geordert werden (C-Compiler, Microsoft Macro Assembler (MASM), Device Driver Kit (DDK), Windows SDK). Das endgültige OS/2 Programmer's Reference Bd. 1-3 und das Buch »Programming the OS/2 Presentation Manager« von Charles Petzold werden ab Juli 1989 vom Vieweg-Verlag über den Buchhandel vertrieben.

Das Betriebssystem selbst (OS/2 PM) müssen Entwickler ab sofort bei den Hardwarelieferanten beziehen, die die OS/2 PM-Version mittlerweile freigegeben haben. Mit den oben genannten vier Herstellern sind bereits etwa 90% aller PCs abgedeckt, die zur Entwicklung von OS/2-Applikationen benutzt werden. Auch der OS/2 LAN-Manager muß jetzt über OEMs (Netzwerklieferanten wie 3Com, Torus, Ungermann-Bass u.a.) bezogen werden. Der im NDK enthalten gewesene SQL-Server wird als Retail-Produkt von Ashton-Tate vertrieben.

Um Programmierer noch besser als bisher zu unterstützen, gibt es von Microsoft ab Juni neue Programmierwerkzeuge für den OS/2 Presentation Manager, die wir im folgenden zusammenfassen wollen:

Presentation Manager Toolkit

Das OS/2 Presentation Manager Toolkit (PTK) Version 1.1 ist der Nachfolger des OS/2 Programmer's Toolkit Version 1.0 und enthält:

- Vier MS-Press-Bücher: Das »MS Press OS/2 Programmer's Reference Vol. 1-3« und »Programming the OS/2 Presentation Manager« von Charles Petzold
- PM SW-Tools (Icon Editor, Dialog Editor, Font Editor...)
- 2,5 Mbyte Sample Code
- Dokumentation auch in Online-Form via QuickHelp.

Das PTK 1.1 kostet 1080 DM zzgl. MwSt. Das Update von 1.0 kostet 440 DM zzgl. MwSt.

Presentation Manager Softset

Zusätzlich zum PTK gibt es als Untermenge davon ein Produkt namens »Microsoft Presentation Manager Softset 1.1«.

Es enthält ein Buch zur Beschreibung der mitgelieferten Tools, die übrigens mit den Tools des PTK (Icon Editor, Dialog Editor usw.) identisch sind. Zusätzlich sind 200 Kbyte Beispielcode enthalten. In Verbindung mit einer

Mitteilungen .... Software .... Mitteilungen

OS/2-Programmiersprache hat man damit alle Software, die zur Anwendungsprogrammierung für OS/2 notwendig ist. Gedacht ist das Softset als Begleitsoftware für die Microsoft Press-Bücher (3x Reference + Petzolds Buch), die auch unabhängig vom PTK bestellt werden können. Der Preis liegt bei 450 DM inkl. MwSt.

Beide Produkte (PTK und Softset) sind sowohl auf 3 1/2- als auch auf 5 1/4-Zoll-Disketten verfügbar.

Ganz besonders empfehlenswert ist die CD ROM Programmer's Library, Version 1.1. Sie umfaßt nun die komplette PM-, LAN Man- und SQL Server-Dokumentation auf einer einzigen CD-ROM, dazu die gesamte Dokumentation zu allen OS/2-Compilern, dem Windows SDK, MS-DOS, der Microsoft Maus und der Intel-Hardware-Familie (80x86/88), und kommt mit einer Dreifach-Lizenz für Netzwerkbenutzer und OPTI-NET Software. Jede weitere Benutzung im Netzwerk kostet 120 DM. Der Preis beträgt 845 DM inkl. MwSt. Der Update von Version 1.0 ist kostenlos.

Programmers's Library auf CD-ROM wesentlich erweitert

Die Version 1.1 der Microsoft CD-ROM Programmers's Library, die Microsoft in diesen Tagen vorstellte, wurde wesentlich erweitert: Auf einer einzigen Compact Disk stehen Ihnen damit insgesamt 72 Bücher und mehr als 26 Mbyte an Beispielprogrammen zur Verfügung. Der Inhalt der CD wurde damit gegenüber der Vorgängerversion um mehr als 225% vergrößert. Darüber hinaus gibt es eine erweiterte Suchfunktion, mit der sowohl in einem speziellen Buch, als auch in allen in einem der Verzeichnisse enthaltenen Bücher nach Informationen gesucht werden kann.

Die Microsoft CD-ROM Programmer's Library 1.1 ist die zeitgemäße Informationsquelle für den professionellen Programmierer in der MS-DOS- beziehungsweise in der OS/2-Welt. Sie beinhaltet umfassende Informationen über MS OS/2, Windows, den Microsoft LAN-Manager, MS-DOS, die Microsoft-Programmiersprachen C, Macro Assembler, Basic, Pascal und Fortran, sowie der PC-Hardware. Der OS/2-Teil wurde erweitert und enthält nun die MS OS/2-Handbücher für Software-Entwickler einschließlich der Programmierhandbücher zum Presentation Manager.

Der MS-DOS-Teil umfaßt auch Dokumentationen zur neuen Version 4.0 des Betriebssystems. Im Netzwerkteil finden Sie Dokumentationen über den Microsoft LAN-Manager sowie den Microsoft SQL-Server. Neben 13 Ausgaben des Microsoft System Journals (US-Version) enthält die CD auch noch vier Intel-Programierhandbücher zu den Intel-Prozessoren 80286, 80386, 80287, 80387. Insgesamt elf Bücher aus dem Microsoft Press Verlag, darunter »The New Peter Norton Programmer's Guide to the IBM PC and PS/2« von Peter Norton und Richard Wilton, »Advanced OS/2 Programming« von Ray Duncan und »Programming the OS/2 Presentation Manager« von Charles Petzold.

Mehr als 100.000 Querverweise erleichtern das Auffinden sich gegenseitig ergänzender Informationen, auch wenn diese in ganz verschiedenen Kapiteln eines Buches zu finden sind. Markierungen, die jeder Benutzer selbst setzen und auch wieder entfernen kann, erleichtern das Erkennen genutzter Informationen.

Um die Microsoft CD-ROM Programmer's Library Version 1.1 nutzen zu können, ist ein IBM PC, XT, AT, PS/2 oder dazu kompatibler Personalcomputer, der mit MS-DOS, Version 3.1 oder höher, sowie 640 Kbyte RAM ausgestattet ist, nötig. Microsoft empfiehlt zusätzlich die Verwendung eines PCs mit einer Festplatte. Darüber hinaus benötigt der Anwender ein CD-ROM-Laufwerk, das in der Lage ist, Compact Disks im Format ISO 9660 zu lesen und die Microsoft MS-DOS CD-ROM-Extension 2.0. Dieses Programm muß vom Laufwerkhersteller auf das jeweilige Laufwerk angepaßt sein und kann entsprechend nur über diesen Hersteller bezogen werden.

Für den Einsatz in vernetzten Mehrplatzsystemen ist die Version 1.1 besonders kostengünstig. Obwohl der Preis gegenüber der Version 1.0 nur geringfügig angehoben wurde, beinhaltet die neue Version 1.1 (Preis: DM 845 zzgl. MwSt.) eine lizensierte Drei-Benutzer-Version von Opti-Net, die es maximal drei Anwendern gleichzeitig erlaubt, eine CD zu nutzen.

Communications Server komplettiert die Betriebssystemumgebung MS OS/2

Die mit dem Einsatz von Computern zusammenhängenden Investitionen und Kosten haben heute vielfach eine Größe erreicht, die eine bessere Nutzung der DV-Ressourcen fordert. Ein wichtiger Schritt zur Nutzenoptimierung ist die Vernetzung von Arbeitsplatzcomputern durch lokale Netzwerke. In größeren Organisationen und Unternehmen besteht jedoch die Notwendigkeit, PC-Netzwerke auch an Zentral- und Abteilungscomputer anzuschließen, um eine Gesamtvernetzung zu erreichen.

Hierfür haben Microsoft und Digital Communications Associates (DCA) zwei neue Netzwerk-Produkte entwickelt: DCA/Microsoft Communications Server (Comm Server) und DCA/ Microsoft Communications Workstation (Comm Workstation). Beide Produkte unterstützen IBM-Protokolle und Programmierschnittstellen. Sie erweitern die Anschlußmöglichkeiten von PCs und lokalen Netzwerken hin zu Großbereichs-Netzwerken.

Comm Server ist eine Systemsoftware innerhalb der Microsoft LAN-Manager-Familie. Mit ihr kann der Arbeitsplatzcomputer mit IBM-Großcomputern und anderen EDV-Systemen kommunizieren. Wie die anderen Netz-

Mitteilungen .... Software .... Mitteilungen

werkprogramme von Microsoft läuft auch der Comm Server unter MS OS/2 auf der Server-Station und unterstützt sowohl MS-DOS- als auch MS OS/2-PCs.

Die Einzelbenutzer-Konfiguration Comm Workstation läuft auf einer Workstation unter Microsoft OS/2, wobei Arbeitsplatz- und Server-Funktionen quasi vereint sind.

Comm Server und Comm Workstation bieten beide die wesentlichen Funktionen, die heute in einem WAN (Wide Area Network) benötigt werden, um mit komplexen Applikationen arbeiten zu können und um die nächste Generation unternehmensweiter DV-Systeme mit Hilfe von Großbereichs-Netzwerken aufzubauen. Diese Funktionen und Merkmale umfassen:

- 3270-Terminal-Emulation.
- Drucker-Emulation sowie File-Transfer zwischen Arbeitsplatzcomputern und SNA-Großcomputern bzw. Applikationen auf Großcomputer-Basis.
- IBM-kompatible Programmierschnittstellen für MS-DOS und MS OS/2, einschließlich der APPC-Programmierschnittstelle mit LU 6.2 - zur Nutzung und Entwicklung von Peer-to-Peer-Applikationen, die mit SNA-Großcomputern, Minicomputern oder PCs kommunizieren können - sowie EHLLAPI- und SRPI-Programmierschnittstellen mit LU 2 - zum Einsatz und zur Entwicklung von Applikationen, die mit Systemen auf Mainframe-Basis kommunizieren können.
- Eine breite Palette von Standard-Anschlußprotokoll-Optionen, einschließlich SDLC, DFT, 802.2 (Token Ring) und X.25/QLLC.
- Offengelegte Schnittstellen-Spezifikationen für Treiber zur Unterstützung von Adapterkarten beliebiger Hersteller.
- Net-View Interface.

Mit dem Einsatz der neuen Netzwerk-Softwareprodukte von Microsoft und DCA (Digital Communications Associates) sind eine Reihe von Vorteilen verbunden, die sich durch höhere Effizienz und verringerte Kosten bemerkbar machen:

- Geringerer Speicherbedarf und verminderte Verarbeitungsanforderungen beim PC.
- Besserer Einsatz der Host-Ressourcen durch variable Nutzung der Logical Units (LU).
- Gemeinsame Nutzung der Adapterkarten durch alle Workstations.
- Einfache Verwaltung des gesamten Netzes von einer beliebigen Workstation aus.
- Einsatz sowohl von MS-DOS als auch MS OS/2 Workstations. Bei sukzessiver Umstellung von MS-DOS auf MS OS/2 stehen von Anfang an alle Möglichkeiten des Comm Servers für sämtliche verbleibenden MS-DOS Workstations zur Verfügung.

Die Version 1.0 des DCA/Microsoft Comm Servers wird im vierten Quartal 1989 für Endkunden verfügbar sein.

Desktop Presentation für den Apple Macintosh mit Microsoft PowerPoint Version 2.0

Ab sofort liefert Microsoft die deutsche Version 2.0 von Microsoft PowerPoint, der führenden Software zur Erstellung von Präsentationsgrafiken für den Apple Macintosh, aus.

Microsoft PowerPoint ist eine Synthese aus Textverarbeitung und Grafikprogramm, speziell entwickelt, um schnell und auf äußerst einfache Weise Präsentationsunterlagen zu erstellen. Eine Beispielpräsentation verdeutlicht, wie schnell man mit PowerPoint ansprechende Vortragsunterlagen erstellen kann: Nach dem Starten von PowerPoint ist zuerst das »Master«, d.h. die Vorlage zu definieren. Diese Vorlage ist häufig ein Rahmen, der ein Firmenlogo beinhaltet oder den Namen des Präsentierenden. Der Rahmen kann in verschiedenen Formen und Strichstärken erstellt werden. Anschließend erfolgt die Präsentationserstellung, das eigentliche Beschreiben der Folien. Dazu ist in Microsoft PowerPoint eine sehr komfortable Textverarbeitung integriert, die es erlaubt, unterschiedlichste Schriftsätze in verschiedenen Schriftgrößen zu wählen. Selbstverständlich kann auch ein Zeilenlineal eingeblendet und komplexere Formatierungen mit unterschiedlichen Textanordnungen vorgenommen werden.

Die Textverarbeitung stellt dem Anwender darüber hinaus einige Werkzeuge zur Verfügung, die die Arbeit wesentlich erleichtern. Dazu gehört eine Funktion »Suchen und Ersetzen« und für falsch geschriebene Wörter eine deutsche Rechtschreibhilfe mit 80.000 eingetragenen Begriffen. Die Datei, die diese Begriffe speichert, kann jederzeit mit häufig benötigten Wörtern ergänzt werden.

Nachdem die gesamten Folien erfaßt wurden, kann die Präsentation mit Hilfe einer mitgelieferten Symbolbibliothek noch etwas »aufgelockert« werden. Dazu stehen in einer Datei einige Dutzend vorgefertigte Pfeile und Grafiken zur Verfügung.

Microsoft PowerPoint bietet außerdem die Möglichkeit, Diagramme oder Grafiken, die mit anderen Programmen (z.B. Microsoft Excel) erstellt wurden, in die Folien zu übernehmen. Dies kann wichtig sein, wenn Zahlenmaterial wie Verkaufs- oder Umsatzstatistiken präsentiert werden sollen.

Die »Slide Show«-Funktion von Microsoft PowerPoint ermöglicht einen Durchlauf aller erstellten Folien am Bildschirm. Sollte der Benutzer dabei feststellen, daß eine andere Reihenfolge der Vortragsunterlagen günstiger wäre, kann die Umstellung ganz einfach mit einer der beiden Sortierfunktionen vorgenommen werden: PowerPoint zeigt wahlweise Titel oder Miniaturabbildungen der Folien, die dann beliebig verschoben werden können.

Mit der Version 2.0 bietet Microsoft PowerPoint nun noch einige neue Features. Das wichtigste ist wohl die volle Farbunterstützung des Macintosh II. Dazu hat Microsoft in Kooperation mit dem größten Dienstleistungsunternehmen

Mitteilungen .... Software .... Mitteilungen

für Präsentationslichtbilder, der Firma Genigraphics, ein Colormodul in PowerPoint integriert, das alle auf dem Mac II darstellbaren 16,8 Millionen Farben zur Verfügung stellt.

Microsoft hat die Erfahrung von Genigraphics genutzt und PowerPoint so ausgelegt, daß der Anwender nach Wahl seiner Hintergrundfarbe automatisch jeweils acht Vorschläge für die Vordergrundfarbe erhält. Natürlich lassen sich nach wie vor die Vorschläge »ablehnen« und eigene Farben anhand eines Farbenkreises generieren.

Die Möglichkeiten einer individuellen farbigen Gestaltung sind damit nahezu grenzenlos. Hier noch zwei Beispiele aus den umfangreichen neuen Leistungsmerkmalen von Microsoft PowerPoint: Der Hintergrund von Folien oder Dias läßt sich mit Microsoft PowerPoint 2.0 stufenlos schattieren und zwar von fast allen Richtungen aus. Eine zweite Neuerung ist die Möglichkeit, einfarbige Grafiken nachträglich zu kolorieren. Mit dem Befehl »Bild kolorieren« läßt sich jeder Grauton der Grafik durch eine Farbe ersetzen.

Microsoft PowerPoint Version 2.0 ist ein spezielles Softwareprogramm für den Apple Macintosh und für diejenigen, die hohe Anforderungen an die Erstellung einer perfekten Präsentation stellen. Die Version 2.0 von Microsoft PowerPoint kostet 1.297 DM (inkl. MwSt).

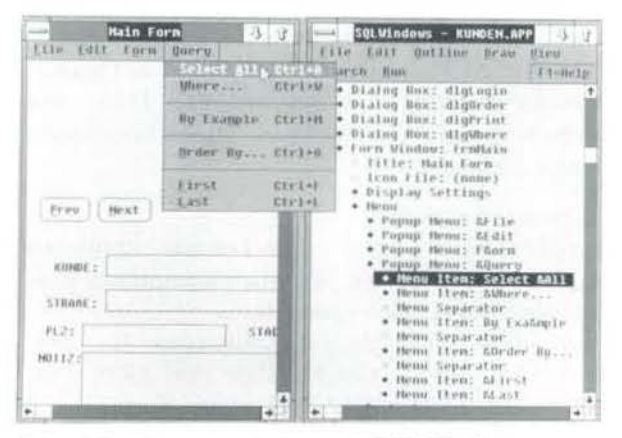
Microsoft richtet Europazentrale in Paris ein

Rund die Hälfte ihres gesamten Umsatzes (Geschäftsjahr 1987/88: 590,8 Mio US$) tätigt die Microsoft Corp. gegenwärtig bereits außerhalb der USA, wobei das Europa-Geschäft mit einem derzeitigen Anteil von über 30% zunehmend an Bedeutung gewinnt. Eine Entwicklung, die das Unternehmen künftig weiter forcieren wird und der man jetzt mit der Etablierung einer Europazentrale in Paris Rechnung getragen hat. An die Spitze der europäischen Zentrale berief Microsoft in diesen Tagen Bernard Vergnes als Vice President Europe, der seit 1983 die französische Niederlassung des US-Softwarespezialisten führte.

Wie sehr Microsoft die Entwicklung in Europa im Auge hat, unterstreichen auch die Berufungen von Jeremy Butler als Senior Vice President für das internationale und OEM-Geschäft sowie von Joachim Kempin als Vice President für OEM-Verkauf und Product Support-Services. Butler zeichnete zuletzt als General Manager of European Sales verantwortlich. Joachim Kempin hat in der Bundesrepublik die starke deutsche Microsoft-Präsenz aufgebaut, ehe er 1987 zur Muttergesellschaft nach Redmond wechselte. Christian Wedell, Geschäftsführer der Microsoft GmbH: »Besonders unsere Anwender werden von der Zusammenführung der Support-Aktivitäten auf weltweiter Ebene profitieren.« Die Verantwortung für die Microsoft-Aktivitäten in der BRD, Schweiz und Österreich verbleiben weiterhin bei der Microsoft GmbH, München-Unterschleißheim.

Siemens übernimmt SQLBase und SQLWindows und Gupta Technologies

Gupta Technologies und die Siemens AG haben vereinbart, daß Siemens ab sofort Guptas SQLBase Server und SQL-Windows Software (MS Windows-Oberfläche) für Einplatzsysteme und vernetzte PC-Lösungen vermarkten wird.



Interaktive Programmierung mit SQLWindows

Der SQLBase Server ermöglicht die Entwicklung von Datenbank-Anwendungen mit Multi-User-Eigenschaften auf PC-Netzen, die früher weit größere Systeme erforderten. Das Produkt, vorgestellt im Oktober 1986, ist derzeit der einzig verfügbare Datenbank-Server, der sowohl unter MS-DOS als auch unter OS/2 und in allen wichtigen PC Netzwerken läuft. Anwendungen, die mit SQLBase oder SQLWindows entwickelt wurden, lassen sich auf Einzelplatzsystemen genauso wie auf vernetzten PCs einsetzen. Gleichzeitig ermöglichen sie den Zugang zu IBMs Datenbank DB2 und Großrechnern.

Die Software von Gupta wird in die Comfoware-Produktpalette (Vertrieb über SPI) von Siemens integriert, bei der unterschiedliche Anwendungsprogramme mit grafischen Bedieneroberflächen ein für den Benutzer außerordentlich komfortables Bürosystem auf PC-Basis bilden. Zusammen mit dem Hochleistungsnetz Comfonet/S ergibt sich mit dem SQLBase Server von Gupta Technologies ein leistungsfähiges System für verteilte Datenbank-Anwendungen in einem PC Netz.

Gupta Technologies ist eines der führenden Unternehmen im wachsenden Markt der netzwerkorientierten, unternehmensweiten Datenverarbeitung (cooperative processing). Guptas direkte Vertriebspartner sind Fortune 1000 Unternehmen, VARs und OEM-Kunden. Guptas Vertriebsorganisation umfaßt neben eigenen Büros in den USA auch Vertragspartner in Europa; so verfügt Gupta über Distributoren in England, Frankreich, der Schweiz sowie der Bundesrepublik Deutschland. Die wichtigsten OEM- bzw. Distributionspartner sind Nokia Information Systems in Skandinavien und Mitsubishi Corporation in Japan.

Mitteilungen .... Software .... Mitteilungen

MS LAN-Manager unter OS/2 im Powernetz von Schneider & Koch

Der LAN-Manager von Microsoft feierte bei Schneider & Koch Premiere. Zum ersten Male zeigten die Netzwerker aus Karlsruhe auf der CeBIT '89 die Implementierung des neuen LAN-Betriebssystems von Microsoft auf dem Hochleistungsnetzwerk SK-NET der breiten Öffentlichkeit.

Nachdem die Testphase mit mehreren Pilotinstallationen in Systemhäusern und Entwicklungsabteilungen großer Konzerne erfolgreich abgeschlossen wurde, stehen den OS/2-Entwicklern und -Anwendern lauffähige Systeme zur Verfügung. Zur Steigerung der Performance wurden neue OS/2- und DOS-Treiber von Schneider & Koch entwickelt und in das SK-LAN-Manager Serverkit implementiert.

Als weitere Produktinnovation von Schneider & Koch wurde das von SCO und Microsoft entwickelte XENIX-NET auf den LAN-Manager im SK-NET adaptiert. Mit diesem SK-XENIX Server-Interface ist es jetzt möglich, Rechner unter XENIX, OS/2 und DOS mit Hilfe des SK-LAN-Managers in einem Netzwerk zu vereinen. Somit stehen den unter dem SK-LAN-Manager laufenden PCs entfernte XENIX-Ressourcen ebenso zur Verfügung, wie normalerweise entfernte Speicherkapazitäten oder Drucker auf anderen DOS- und OS/2-Maschinen. Aufgrund der bereits vorhandenen großen Anzahl von Applikationen unter XENIX, wird die XENIX/UNIX-Integration in ein zentrales Ethernet-LAN an Bedeutung gewinnen.

Schneider & Koch sieht in der Kooperation mit Microsoft eine Möglichkeit, High Tech made in Germany innerhalb der OS/2-Welt auszubauen. Multikfunktionale Kommunikationsprotokolle werden die Connectivity in einem InterNet von Schneider & Koch enorm erweitern.

Der Vorteil für den Anwender liegt darin, daß praktisch von allen Workstations im Netzwerk gleichzeitig auf die gleichen Daten im Server zugegriffen werden kann. Zusätzlich wird diese Netzwerk-Datenbank von den Möglichkeiten des erweiterten Datenschutzes und der leichteren Verwaltung, wie sie der OS/2 LAN-Manager bietet, unterstützt.

3Com-Netzwerk jetzt auch mit SQL-Server

Der kalifornische Netzwerk-Hersteller 3Com gibt die Freigabe der SQL-Server-Version von Ashton-Tate für 3Com-Netzwerke unter dem »3+Open LAN Manager« bekannt. Damit können 3+Open-Anwender auch komplexe Datenbank-Probleme in einem PC-Netzwerk lösen. Anwendungen, wie Ticket-Reservierungen bei Fluglinien und Auftrags-Abwicklung in größeren Unternehmen sind Beispiele dafür.

Der SQL-Server ist eine relationale Datenbank, die von Ashton-Tate und Microsoft gemeinsam speziell für OS/2-Anwender entwickelt wurde. Seine Entstehung verdankt dieses Produkt nicht zuletzt den technischen Möglichkeiten, die der OS/2-LAN Manager in Form von APIs (Application Program Interface) anbietet. In der offenen Netzwerk-Architektur, wie sie der »3+Open LAN Manager« anbietet, verhält sich der SQL-Server, wie jeder andere Netzwerk-Dienst. Er nutzt dabei die Teilung der Aufgaben zwischen Server und Workstation.

---

db-Vista III jetzt auch für MS-Windows

Ab sofort ist db-Vista III, die Datenbank für C-Programmierer, auch für MS-Windows erhältlich. Damit können C-Programmierer erstmals die Vorteile einer schnellen und flexiblen Datenbank auf Netzwerkbasis mit der Windows-Philosophie verbinden. db-Vista folgt mit seinem Release 3.1 den Windows-Level-II-Regeln. Die mit Windows/db-Vista III entwickelte Applikation besteht aus kleinen Einzelprogrammen, die jeweils im Medium Memory Model kompiliert werden. Den einzelnen Modulen stehen jeweils volle 64 Kbyte im Datensegment zur Verfügung, da db-Vista III seinen eigenen Speicherbedarf aus dem Windows Global Memory deckt. Die einzelnen Module oder Tasks sind jeweils in einem eigenen Fenster darstellbar - in der Multiuser-Version kann so über mehrere Fenster auf dieselbe Datenbank zugegriffen werden. Der Vorteil liegt auf der Hand: Während in einem Fenster z.B. ein neuer Kunde eingegeben wird, kann in einem anderen Fenster eine Liste der bereits bestehenden Kunden angezeigt werden. Die mit db-Vista III entwickelten Applikationen sind voll in die Windows-Philosophie integrierbar und können leicht auf OS/2 und den Presentation Manager portiert werden.

Durch den Swapping-Mechanismus von Windows gibt es keine Einschränkung der Applikationsgröße. db-Vista für Windows ist in einem Netzwerk oder auf Einzelrechnern lauffähig. Ein mitgelieferter NetBios-Emulator sorgt dafür, daß die Multiuser-Version von db-Vista auch in einer Umgebung ohne Netzwerk lauffähig ist.

Norton Utilities 4.5 in deutsch

Der Markt & Technik Verlag kündigte für Anfang Juni die deutsche Version 4.5 der Norton Utilities an. Die neue Version der Norton Utilities ist weit mehr, als eine überarbeitete Version des Vorgängers. Durch den vollautomatischen Disketten-Doktor und das Norton-Kontrollzentrum setzt diese Utility-Sammlung neue Maßstäbe. Neben der extremen Leistungssteigerung gibt es vor allem für die Benutzeroberfläche einige Änderungen zu berichten. Trotz der reinen ASCII-Oberfläche hat man bei der Bedienung der neuen Utilities das Gefühl, daß hier die »Maus-Ziel-Klick-Macintosh-Oberfläche« Pate gestanden hat. Auf jeden Fall sind die neuen Norton Utilities die unerläßliche Hilfe für PC-Benutzer in allen Lebens- und Datenlagen.

Mitteilungen .... Hardware .... Mitteilungen

Compaq mit neuem 33-MHz 80386-PC

Compaq brachte unlängst drei Modelle des Compaq Deskpro 386/33 auf den Markt. Er ist einer der derzeit weltweit leistungsfähigsten und erweiterbarsten Personalcomputer.



Der Deskpro 386/33 hat sich äußerlich gegenüber seinem 25-MHz-Kollegen kaum verändert, doch bescheinigt ihm Compaq 35% mehr Leistungsfähigkeit.

Der Compaq Deskpro 386/33 ist für anspruchsvolle Anwendungen ausgelegt, wie z.B. Software-Entwicklung, CAD, Finanzanalysen sowie für den Einsatz als File-Server in Multi-User-Systemen. Dieser Computer basiert auf der Compaq Flex-Architektur und erreicht durch Kombination des fortschrittlichsten 386-Prozessors, des 33-MHz-80386 von Intel, mit einem 64 Kbyte Cache-Speicher bei rechenintensiven Anwendungen eine um 35 Prozent höhere Leistung als Personalcomputer mit 25 MHz Taktrate, 80386-Prozessor und Cache-Speicher.

Der Compaq Deskpro 386/33 zeichnet sich durch platzsparendes ergonomisches Design aus und bietet acht Erweiterungs-Steckplätze sowie Raum für den Einbau von fünf Massenspeichern, die es ermöglichen, die interne Massenspeicherkapazität auf ein Maximum von 1,3 Gigabyte aufzurüsten - mehr als jeder andere derzeit erhältliche Desktop Personalcomputer. Ein 32-Bit-Steckplatz ermöglicht das Aufrüsten des Systems auf bis zu 16 Mbyte 32-Bit RAM.

Der Compaq Deskpro 386/33 bietet kaum erreichte Verarbeitungsgeschwindigkeit und Leistung. Durch Kombination der Compaq Flex-Architektur mit der höheren Taktrate des 33-MHz-Prozessors ergeben sich beim Compaq Dektop einige Leistungsvorteile gegenüber anderen Systemen. Die Flex-Architektur ermöglicht durch das »Concurrent-Bus-Prinzip«, bei dem Speicher- und Peripheriebus gleichzeitig angesprochen werden, eine gleichmäßige Auslastung und Optimierung aller Subsysteme.

Die Flex-Architektur ist mit einem 32-Bit-Speicherbus ausgestattet, der Speicherzugriffe bei voller Taktrate ermöglicht und 98 Prozent der Zeit mit Null Waitstates arbeitet. Ein separater Industriestandard-Erweiterungsbus für Peripheriegeräte ermöglicht die Verwendung von 8-Bit- und 16-Bit-Erweiterungsplatinen und Peripheriegeräten für PCs im Industriestandard.

Der Compaq Deskpro 386/33 ist mit getrennten Sockeln für Coprozessoren ausgestattet, die den gleichzeitigen Einbau und Betrieb der neuen Coprozessoren 33-MHz Intel 80387 und 33-MHz Weitek 3167 ermöglichen. Der 33-MHz Weitek 3167 Coprozessor ermöglicht bei besonders anspruchsvollen wissenschaftlichen, konstruktionstechnischen und finanzanalytischen Anwendungen eine zwei- bis dreimal höhere Leistung als der 33-MHz-80387-Coprozessor.

Durch Integration des VG-Controllers und der Standard-Schnittstellen auf der Systemplatine bleiben Erweiterungs-Steckplätze für andere Verwendungszwecke frei. Der Compaq Deskpro 386/33 bietet bessere Erweiterungsmöglichkeiten und größere Speicherkapazität als andere 80386 Dektop-PCs, einen 32-Bit Steckplatz für bis zu 16 Mbyte RAM, fünf 8-/16-Bit Erweiterungs-Steckplätze, einen 8-Bit Steckplatz und Raum für fünf interne Massenspeicher.

Auf den Modellen des Compaq Deskpro 386/33 können Tausende von MS-DOS-, Microsoft Windwos/386- und UNIX/XENIX-Anwendungen sowie neue MS OS/2-Anwendungen für PC im Industriestandard ablaufen.

Arbeitsspeicher

Der Compaq Deskpro 386/33 verfügt über einen 32-Bit-Erweiterungs-Steckplatz, der ein Aufrüsten des Systems auf bis zu 16 MByte 32-Bit-RAM ermöglicht. Der 80385-Cache-Controller von Intel verwaltet den 64 Kbyte Cache-Speicher, der eine Zugriffszeit von 25 Nanosekunden hat und es dem System ermöglicht, bei fast allen Speicherzugriffen mit der vollen Taktrate von 33 MHz und über 98 Prozent der Aktivzeit mit Null Waitstates zu arbeiten.

Der Compaq Expanded Memory Manager (das Dienstprogramm CEMM) nutzt die Vorteile der Anwendungen, die der LIM/EMS-Spezifikation (Lotus/Intel/Microsoft 4.0 Expanded Memory Specification) entsprechen und den Zugriff auf Dateien im Speicherbereich jenseits der MS-DOS-Grenze von 640 Kbyte ermöglichen.

Massenspeicher

Der Compaq Deskpro 386/33 ermöglicht den internen Einbau von bis zu fünf Massenspeichern. Dadurch kann die interne Speicherkapazität auf bis zu 1,3 Gigabyte erweitert werden, mehr, als jeder andere Desktop Computer derzeit bieten kann. Alle Festplattenlaufwerke arbeiten mit Interleave-Faktor 1:1 und ermöglichen dadurch schnellen Datenzugriff. Die 84-Mbyte-Festplatte hat eine mittlere Zugriffs-

Mitteilungen .... Hardware .... Mitteilungen

zeit von weniger als 25 Millisekunden, die 320- und 650-Mbyte Festplatten haben eine mittlere Zugriffszeit von weniger als 18 Millisekunden.

Video

Der Compaq Deskpro 386/33 bietet integrierte VGA-Grafik, die zum Industrie-Standard voll kompatibel ist, jedoch aufgrund des 16-Bit-Datenpfades eine um 50 Prozent höhere Leistung erreichen soll als VGA-Grafik-Systeme anderer Hersteller.

Grundausstattung

Der Compaq Deskpro 386/33 ist standardmäßig mit einem VG-Controller und Schnittstellen für schnelle asynchrone Datenübertragung, Zeigegeräte, parallele Datenübertragung und Druckeranschluß sowie hochleistungsfähigen Festplattenlaufwerken ausgestattet.

Ebenfalls zur Standardausrüstung gehören acht Erweiterungs-Steckplätze voller Länge (sieben Steckplätze im Industrie-Standard und ein 32-Bit RAM Hochgeschwindigkeits-Steckplatz); 300-Watt-Netzteil mit automatischer Spannungsumschaltung; die Dienstprogramme CEMM und CACHE sowie eine Garantie auf ein Jahr.

Alle Modelle des Compaq Deskpro 386/33 sind standardmäßig mit 2 Mbyte 32-Bit RAM und einem 5 1/4-Zoll Diskettenlaufwerk (1,2 Mbyte) ausgestattet. Das Modell 84 verfügt über eine 84-Mbyte Festplatte halber Bauhöhe. Die Modelle 320 und 650 sind mit einem 320-Mbyte Festplattenlaufwerk halber Bauhöhe bzw. mit einem 650-Mbyte Festplattenlaufwerk voller Bauhöhe einschließlich zugehörigem 15-MHz ESDI-Controller ausgestattet.

Scanner mit hoher Auflösung

Die Scannerfamilie von Siemens wurde durch die Modelle High-Scan 600 und HighScan 800 erweitert. Ihre höhere Auflösung bringt insbesondere bei graphischen Anwendungen und bei der Schrifterkennung Vorteile. Mit den Modellen der HighScan-Familie will sich Siemens an die Spitze der Entwicklung bei professionellen Desk-Top-Scannern setzen. Mit der erreichten Auflösung lassen sich selbst reprofähige Vorlagen im Desktop-Publishing erstellen.

Ein weiterer Einsatzschwerpunkt der hochauflösenden HighScan-Modelle ist die optische Zeichenerkennung (OCR). So lassen sich schon mit einer physikalischen Auflösung von 400 dpi etwa auch Petitschriften im 6-Punkt-Bereich ohne Einschränkungen verarbeiten, z.B. bei der lernfähigen OCR-Software OPTOPUS. Im Gegensatz zu 300-dpi-Scannern werden auch bei Vorlagen mit Schriftmischungen die mit HighScan eingelesenen Zeichen meist bereits nach einem einmaligen Lernvorgang sicher wiedererkannt. Auch bei einer vortrainierten OCR-Software, wie z.B. RECOGNITA, zeigt sich, wie sehr es beim Einlesen auf eine möglichst hohe Auflösung ankommt.

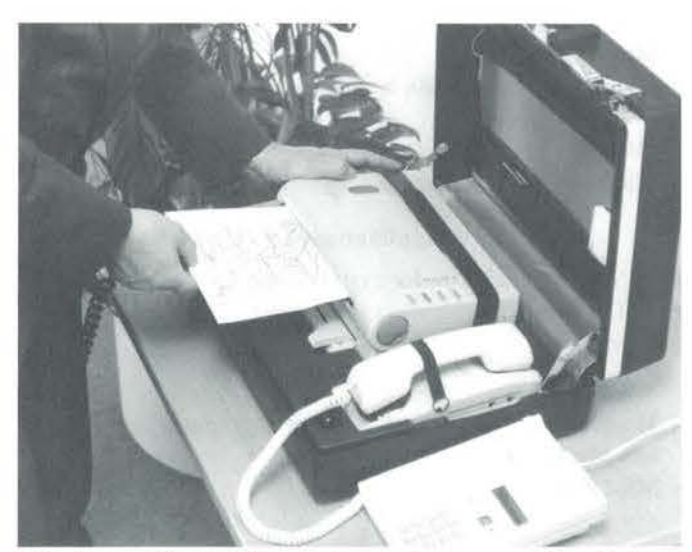
AST baut 80486-Rechner

AST Research Inc., Irvine, Kalifornien/USA, entwickelt bereits einen 80486er Rechner, der die bisherige 286 und 386er Reihe als Topend erweitern soll.

Als Haupteinsatzgebiet werden komplizierte und rechenintensive Anwendungen gesehen wie beispielsweise Modellentwicklung, Grafikerstellung, CAD/CAM oder Finanzanalysen. Von diesem Rechnertyp wird erwartet, daß er die nächste Dekade beherrschen wird. Inwieweit er eine direkte Konkurrenz zu Minicomputern sein wird, bleibt abzuwarten. AST wird diesen Rechner mit dem Intel-Microprozessor ausrüsten, und außerdem mit der neuen EISA-Bauweise (Erweiterte Industrie Standard Architektur) versehen. Die Auslieferung der neuen Computer ist für das zweite Quartal 1990 vorgesehen.

Faxen - überall, wo es ein Telefon gibt

Für das Fernkopieren unterwegs wurde auf der Basis des Siemens Minifax das Mobilfax HF2301M entwickelt. Als Koffergerät mit integriertem Akustikkoppler CTK Speedy-Fax 9600 ist das Mobilfax überall nutzbar, wo ein Telefon und ein Stromanschluß zur Verfügung steht, z.B. im Hotelzimmer oder zu Hause.



Mit der mobilen Kofferausrüstung kann man überall faxen, wo sich ein Telefon und eine Steckdose in der Nähe befinden.

Mit dem HF 2301M - ein Gruppe-3-Fax - sind mit 9600 bit/s Übertragungszeiten von unter einer halben Minute pro Normseite erzielbar. Werden Fotos detailgetreu in 16 Graustufen umgesetzt, dauert es entsprechend länger. Im Stapelbetrieb können bis zu fünf Seiten automatisch an das angewählte Faxgerät kopiert oder als Lokalkopie gezogen werden. Für den Empfang von Fernkopien steht ein Papiervorrat von rund 100 Seiten zur Verfügung.

Termine ... Termine ... Termine ... Termine

Mit Microsoft-Seminaren sicher in die Zukunft

Die ständige Leistungssteigerung in der Hard- und Software-Entwicklung läßt den Fachkräften der Industrie kaum eine Chance, die Möglichkeiten dieser Technologie in gleicher Schnelligkeit umzusetzen. Dabei ist es von sehr großer Bedeutung, Arbeitsbereiche mit Hilfe der EDV zu rationalisieren. Um die Integration der EDV und damit eine Arbeitserleichterung im betriebswirtschaftlichen Alltag zu erreichen, wird Microsoft die Institut-Seminare in München und Düsseldorf fortsetzen.

Diese Spezialseminare des Microsoft Instituts vermitteln in kleinen Gruppen intensiv all das, was zum Einstieg in die Programmentwicklung nötig ist. Modernste Trainingsmethoden sowie PC-Demonstrationen und -Übungen sind selbstverständlich. Die Dozenten befassen sich auch im persönlichen Gespräch ausführlich mit den individuellen Forderungen und Problemen der Teilnehmer. So bekommen professionelle Entwickler durch professionelle Schulung die Möglichkeit, ihren hohen Wissenstand den neuen Gegebenheiten anzupassen. Das Microsoft Institut bietet Seminare in drei Bereichen an:

Systemsoftware

Diese Seminare sind für PC-Software-Entwickler bestimmt. Die angebotenen Themen:

- Microsoft OS/2
- Microsoft Presentation Manager
- Microsoft Windows
- Microsoft LAN Manager
- Microsoft SQL Server

Applikationen

Diese Seminare sind in der ersten Linie für die Mitarbeiter des EDV-Benutzer-Services bestimmt. Die angebotenen Themen:

- MS-Word
- MS-Excel und MS-Excel Makro-Programmierung
- MS-Multiplan
- MS-Chart

Informationsseminare

Diese Seminare sind für die Mitarbeiter des EDV-Benutzer-Service bestimmt. Die angebotenen Themen:

- Einblick in Microsoft System-Software
- LAN Manager und SQL Server

Das Microsoft OS/2 Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Neben einem Überblick über den Intel 80286-Prozessor wird dann als Schwerpunkt dieses Seminars auf das Application Program Interface (API) eingegangen. In diesem Zusammenhang werden die Funktionen für Multitasking, Memory Management und Dynamic Linking ausführlich behandelt. Weitere Themen: Geräte- und Datei-Ein-/-Ausgabe. Die in diesem Seminar gezeigten Programmbeispiele sind in Microsoft C geschrieben.

Ort	Datum	Tag
Düsseldorf	31.07./01.08.89	Mo./Di.
	04./05.09.89	Mo./Di.
München	17./18.07.89	Mo./Di.

Der Microsoft OS/2 Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das MS-OS/2-Einführungsseminar besucht haben.

Die Teilnehmer lernen im Vortrag und praktischen Übungen am PC den komfortablen Editor zu nutzen, Family-API-Programme zu schreiben und Device-I/O-Routinen zu erstellen sowie Multitasking-Funktionen zu nutzen und eigene Dynamic-Link-Bibliotheken zu erstellen; außerdem können sie die erweiterten Speicherverwaltungsmöglichkeiten des Intel 80286 nutzen und mit Hilfe des MS-OS/2 Memory Managers programmieren.

Ort	Datum	Tag
Düsseldorf	02./03./04.08.89	Mi./Do./Fr.
	06./07./08.09.89	Mi./Do./Fr.
München	19./20./21.07.89	Mi./Do./Fr.

Das Microsoft Windows Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Dieses Seminar behandelt die Microsoft Windows-Benutzerschnittstelle, die Microsoft Windows-Systemarchitektur, die Programmierwerkzeuge in der Praxis und Erstellung von Microsoft Windows-Programmen an Beispielen.

Ort	Datum	Tag
Düsseldorf	17./18.07.89	Mo./Di.
	21./22.08.89	Mo./Di.
München	10./11.07.89	Mo./Di.
	04./05.09.89	Mo./Di.

Termine ... Termine ... Termine ... Termine

Der Microsoft Windows Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das Microsoft Windows Einführungsseminar besucht haben.

In diesem Seminar erwirbt der Teilnehmer durch praktische Tätigkeit an einem PC die Fertigkeiten zur Programmierung unter Microsoft Windows. Dazu werden verschiedene Aufgaben gestellt und Übungen abgehalten. Als Hauptthemen dieses Seminars werden Übungen zu dem »Windowing«, der grafischen Programmierschnittstelle, den Benutzerschnittstellen und dem Dynamic Linking durchgeführt.

Ort	Datum	Tag
Düsseldorf	19./20./21.07.89	Mi./Do./Fr.
	23./24./25.08.89	Mi./Do./Fr.
München	12./13./14.07.89	Mi./Do./Fr.
	06./07./08.09.89	Mi./Do./Fr.

Microsoft Presentation Manager Einführungs-Seminar

Dieses zweitägige Seminar für Windows- und C-Programmierer erläutert das Konzept des Presentation Managers und gibt einen Überblick über seine Programmierschnittstellen unf Fähigkeiten.

Das Seminar behandelt die Benutzerschnittstelle des Presentation Managers, seine Architektur, die Programmierwerkzeuge in der Praxis und die Erstellung von PM-Programmen an Beispielen.

Ort	Datum	Tag
Düsseldorf	25./26.09.89	Mo./Di.
München	27./28.07.89	Do./Fr.

Microsoft Presentation Manager Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in C und eventuell in Windows besitzen sowie das Microsoft Presentation Manager Einführungsseminar besucht haben.

In diesem Seminar erwirbt der Teilnehmer durch praktische Tätigkeit an einem PC die Fertigkeiten zur Programmierung unter dem Microsoft Presentation Manager. Als Hauptthemen dieses Seminars werden das Application Program Interface (API) des Presentation Managers besprochen. Dazu werden verschiedene Aufgaben gestellt und Übungen abgehalten: Windowing, Dynamic Linking Libraries usw.

Ort	Datum	Tag
Düsseldorf	27./28./29.09.89	Mi./Do./Fr.

Microsoft LAN Manager Einführungs-Seminar

Dieses zweitägige Seminar für Netzanwender und Netzwerk-Softwareentwickler mit Programmiererfahrung in C vermittelt das neue zukunftsweisende Programmierkonzept, um den MS OS/2 LAN Manager bedienen zu können.

In diesem Seminar werden die neuen Konzepte des LAN Managers erörtert sowie neue und zukunftsweisende Programmiermethoden (hier ist die »distributed intelligence Method« zu nennen) vorgestellt. Weiterhin gibt das Seminar in seinem letzten Abschnitt eine Installations- und Bedienungsanleitung für den LAN Manager und vermittelt so einen »roten Faden« für den Netzwerkanwender.

Ort	Datum	Tag
Düsseldorf	10./11.07.89	Mo./Di.
	28./29.08.89	Mo./Di.
München	03./04.07.89	Mo./Di.

Microsoft LAN Manager Worksop

Dieses dreitägige Seminar für PC-Softwareentwickler mit Programmiererfahrung in C, die am LAN Manager Einführungsseminar teilgenommen haben, vermittelt die Fähigkeit zum Schreiben von Programmen, die die grundlegenden Bestandteile des Microsoft LAN Managers benutzen.

Der Schwerpunkt dieses Seminars liegt auf der Darstellung des API des LAN Managers. Themengebiete wie »Named Pipes«, »Mailslots«, »Server-Client-Konzept« usw. werden mit Hilfe von Übungen und praktischen Programmierbeispielen erläutert.

Ort	Datum	Tag
Düsseldorf	12./13./14.07.89	Mi./Do./Fr.
	30./31.8./1.9.89	Mi./Do./Fr.
München	05./06./07.07.89	Mi./Do./Fr.

Microsoft SQL Server Einführungs-Seminar

Dieses zweitägige Seminar für PC-Softwareentwickler erläutert die Struktur und das Konzept des Microsoft SQL Server. Kenntnisse von MS OS/2 und dem MS LAN Manager sollten vorhanden sein, SQL- und C-Kenntnisse sind von Vorteil. Das Seminar gibt einen Überblick über die Struktur und den Aufbau des Microsoft SQL Servers. Neben der Administration des Systems werden Funktionsweise und Einsatzgebiete des SQL Servers behandelt.

Termine ... Termine ... Termine ... Termine

Ort	Datum	Tag
München	25./26.09.89	Mo./Di.

Microsoft SQL Server Worksop

Dieses dreitägige Seminar für PC-Softwareentwickler mit Kenntnissen in C und SQL, die am Einführungsseminar und Workshop für OS/2 und den LAN Manager teilgenommen haben, vermittelt die Fähigkeit zum Schreiben von Programmen, die die Funktionen des Microsoft SQL Servers benutzen.

Der Schwerpunkt dieses Seminars liegt neben der Installation und Administration des Systems auf der ausführlichen Erläuterung der DB-Library, der Programmierschnittstelle des Systems. Die Methoden zur Programmierung werden mit Hilfe von Übungen und praktischen Programmierbeispielen veranschaulicht.

Ort	Datum	Tag
München	27./28./29.09.89	Mi./Do./Fr.

Microsoft LAN Manager und SQL Server Informationsseminare

Dieses eintägige Informationsseminar wendet sich an Vertriebs- und Supportmitarbeiter sowie Entscheidungsträger, die Gerätekonfigurationen für Arbeitsgruppen festlegen.

Die Teilnehmer gewinnen einen Überblick über die Architektur des Client-Server-Modells und die darin angelegten neuen Möglichkeiten für die Gruppenarbeit in einem Netz durch gemeinsamen Zugriff auf eine zentrale Datenbank.

Dargelegt wird, was den LAN Manager von einem herkömmlichen Netz unterscheidet und welche Vorteile für den Anwender damit verbunden sind.

Die neuartige Architektur des SQL-Servers wird erläutert und welche zum Teil einmaligen, neuen Möglichkeiten sich in Hinsicht auf die Sicherheit, Datenintegrität, Transaktionen und Performance als Plattform für eigene Anwendungen und Verfügbarkeit ergeben. Schließlich werden die Oberflächen des LAN-Managers und des SQL-Servers erklärt und an Beispielen gezeigt, wie man mit dem System arbeitet und welche komplexen Abfragen SQL ermöglicht.

Ort	Datum	Tag
Düsseldorf	22.05.89	Mo.
München	21.06.89	Mi.

Microsoft Excel Makro-Programmierung

Dieses dreitägige Seminar wendet sich an PC-Software-Entwickler, die mit Microsoft Excel schon einigermaßen vertraut sind.

Das Seminar gibt anfangs einen grundsätzlichen Überblick über Microsoft Excel, der für das Verständnis der Makroprogrammierung nötig ist. Durch einfachere Einzelbeispiele wird der Teilnehmer zunächst mit der Makroprogrammierung vertraut gemacht. Es folgt der Einstieg in komplexe Problemlösungen mit Makros, in denen u.a. sowohl selbstdefinierte Menüs und Dialogboxen als auch Funktions- und Befehlsmakros eingebunden sind. Weiterhin geht das Seminar auf die Microsoft Excel-Programmierumgebung ein, wobei auf die Möglichkeiten hingewiesen wird, Informationen aus anderer Software mit Makros zu verarbeiten (Datev Datensätze) und wie mit Hilfe von Makrobefehlen die DDE-Schnittstelle von Microsoft Windows genutzt werden kann.

Ort	Datum	Tag
Düsseldorf	07./08./09.08.89	Mo./Di./Mi.
	12./13./14.09.89	Di./Mi./Do.
München	24./25./26.07.89	Mo./Di./Mi.
	20./21./22.09.89	Mi./Do./Fr.

MS-DOS Systemprogrammier-Seminar für Entwickler

Die Firma Kontron veranstaltet in seinem Schulungscenter in Eching vom 26. bis 30. Juni '89 einen 4-tägigen DOS-Systemprogrammier-Workshop. Es wird die interne Struktur des weitverbreiteten Betriebssystems MS-DOS behandelt. Neben Themen wie

- ROM-BIOS und BIOS-Aufrufe,
- DOS-Funktionsaufrufe,
- Kommando-Interpreter,
- Dateiverwaltung,
- Speicherresidente Programme etc.

wird besonderes Augenmerk auf den Aufbau eines Treibers gelegt. Die Unterschiede zwischen den verschiedenen DOS-Versionen werden aufgezeigt. Dieser Workshop ist für den Personenkreis interessant, der Programme an das Betriebssystem MS-DOS anpassen muß, bzw. Programme für eine MS-DOS-Umgebung entwickelt. Zur Vertiefung des Stoffes steht für je zwei Teilnehmer ein AT-kompatibler Rechner zur Verfügung.

Eine komfortable Benutzeroberfläche in C (Teil 4):

Eine Help-Engine für Ihre SAA-Applikationen

Der elektronische Ratgeber, das Handbuch im Rechner - jederzeit auf Knopfdruck verfügbar - zählt nicht erst seit der Veröffentlichung des SAA-Standards zu den Leistungsmerkmalen, an denen sich jede PC-Applikation messen lassen muß. Da sich viele Programmierer bisher jedoch von der Komplexität eines solchen Hilfe-Moduls abschrecken ließen, möchten wir Ihnen im Rahmen dieser Folge unserer SAA-Serie eine universelle Help-Engine vorstellen, die auf einfachste Art und Weise in jede Applikation eingebunden werden kann.

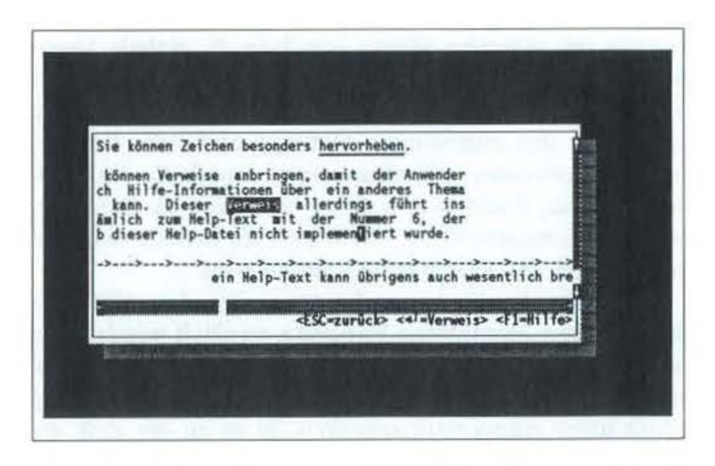


Bild 1: *Das Demo-Programm HLPDEMO.C verdeutlicht die Arbeit mit der Help-Engine.*

Proportional mit der steigenden Leistungsfähigkeit moderner PC-Applikationen ist auch deren Komplexität gewachsen und da der typische PC-Anwender heute mehrere Standard-Applikationen in seiner täglichen Arbeit einsetzt, kann ihm wohl kaum zugemutet werden, daß er alle Befehle, Optionen und Tastenkombinationen der verschiedenen Applikationen jederzeit im Kopf hat. Oft hilft da nur ein Blick in das Benutzerhandbuch weiter, doch lassen sich die benötigten Informationen dort in der Regel nur sehr schwer und unter großem Zeitaufwand finden. Aus diesem Problemfeld heraus wurde die Idee des elektronischen Handbuchs - einer On-Line-Hilfe direkt am Bildschirm - geboren. Das Handbuch im Rechner allein ist jedoch noch nicht der Schlüssel zum Problem, wenn man sich auch innerhalb dieses Handbuchs erst mühsam zur benötigten Information vorarbeiten muß. Effizient wird die Arbeit mit einem solchen Handbuch nämlich erst dann, wenn man es mit einem gewissen Grad an Intelligenz versieht und es dadurch in die Lage versetzt, die aktuelle Situation, in der sich der Anwender befindet, zu erkennen. Dadurch nämlich kann es dem Anwender gezielt Informationen anbieten, die ihm in seiner augenblicklichen Situation weiterhelfen.

Um dies anhand einer konkreten Situation deutlich zu machen: Wenn sich der Anwender beispielsweise innerhalb einer Eingabemaske befindet und über die festgelegte Help-Taste Hilfe anfordert, so ist er bestimmt nicht an Informationen über die verschiedenen Programmbefehle oder die grundsätzliche Funktionsweise des Programms interessiert. Vielmehr ist ein Bezug zum aktuellen Kontext, also zu seiner augenblicklichen Situation innerhalb der Eingabemaske zu vermuten. Angeboten werden sollten ihm deshalb Informationen über die allgemeine Bedeutung der Eingabemaske, die spezielle Bedeutung der einzelnen Felder und Informationen über die Editierung der Maske, d.h. über das Wechseln zwischen den einzelnen Feldern, der Dateneingabe innerhalb dieser Felder etc. Erst dieser Bezug zum aktuellen Kontext, dieses kontextsensitive Verhalten, verleiht einem Hilfe-System seine besondere Benutzerfreundlichkeit.

Ein kontextsensitives Help-System

Um in jeder Situation den passenden Hilfetext parat zu haben, sieht die hier vorgestellte Help-Engine die Unterteilung der Hilfe-Informationen in viele kleine Hilfetexte vor, die jeweils mit einer individuellen Kennummer zwischen 1 und 32.000 versehen werden. Da ein Hilfetext innerhalb einer SAA-Applikation jederzeit durch Betätigung der Taste [F1] abgerufen werden kann, empfiehlt sich die Implementierung einer Help-Routine, die grundsätzlich bei der Betätigung dieser Taste aufgerufen wird. Das Tastatur- und Mausmodul, das in der zweiten Folge dieser Serie vorgestellt wurde, unterstützt die Implementierung einer solchen Routine durch die Funktion `KbdSetFkt()`. Wird diese Funktion am Anfang Ihrer Applikation etwa folgendermaßen aufgerufen:

```
KbdSetFkt( F1, HelpFkt );
```

Dann wird die Funktion `HelpFkt()` mit jeder Betätigung der Taste [F1] aufgerufen. Damit `HelpFkt()` einen kontextbezogenen Hilfetext auf den Bildschirm holen kann, sollte Ihre Applikation über eine globale Variable verfügen, in die die einzelnen Programmteile jeweils die Nummer des ihnen zugeordneten Hilfetexts eintragen.

Beachten Sie bitte auch, daß innerhalb einer SAA-Applikation jedes Hauptmenü und jedes der zugehörigen Untermenüs über einen individuellen Hilfetext verfügen sollte. Während dieser Hilfetext im Falle eines Hauptmenüs nur kurz die Aufgabe dieses Menüpunktes und die Bedeutung der einzelnen Untermenüs beschreibt, müssen die Hilfetexte zu den einzelnen Untermenüs ganz genau auf die Bedeutung des jeweiligen Menüs eingehen. Damit Ihre `HelpFkt()` erkennen kann, ob sich der Anwender innerhalb des Pull-Down-Menüs befindet und wenn ja, welches der Menüs der Menü-Cursor gerade überdeckt, lädt das Menü-Modul (aus der dritten Folge dieser Serie) die Kenn-

nummer des aktuellen Menüs jeweils in die globale Variable aktmen. Mit Hilfe eines Vektors, der die Zuordnungen zwischen Menü- und Hilfetextnummern enthält, können Sie dann leicht den zugehörigen Hilfetext ermitteln. Befindet sich der Anwender allerdings nicht innerhalb des Pull-Down-Menüs, findet sich in dieser Variablen ein Wert, der durch die Konstante KEIN_MENU (aus der Include-Datei für das Menü-Modul MEN.H) repräsentiert wird.

Nachdem die Nummer des anzuzeigenden Hilfetexts über die oben angeführten Variablen ermittelt wurde, kann dieser Hilfetext über eine Funktion der Help-Engine auf den Bildschirm geholt werden. Bevor wir diesem Vorgang jedoch unser Augenmerk schenken, ist es an der Zeit, die Erstellung der einzelnen Hilfetexte in den Mittelpunkt unserer Betrachtungen zu stellen.

Erstellung der Hilfetexte

Voraussetzung für die Anzeige der Hilfetexte ist deren Bereitstellung innerhalb einer unformatierten ASCII-Datei. Zwar finden sich in dieser Datei bereits alle Informationen, die die Help-Engine zur Anzeige der einzelnen Hilfetexte benötigt, doch muß diese Datei aus Performance-Gründen zunächst durch einen Help-Compiler in ein binäres Format gebracht werden. Dieser Help-Compiler (HLPCOMP.EXE), dessen Listing Sie auf den folgenden Seiten finden, nimmt dabei mehrere Aufgaben war. Er arbeitet sich zeilenweise durch die Eingabedatei und führt zunächst eine Kombination aus semantischer und lexikalischer Analyse durch, um syntaktische Fehler innerhalb der Datei zu erkennen. Danach wandelt er die verschiedenen Befehle innerhalb der einzelnen Hilfetexte in ein binäres Format um und widmet sich im letzten Schritt dann seiner umfangreichsten Aufgabe, der Kompression der einzelnen Hilfetexte.

Diese Kompression, die die Größe der entstehenden Help-Datei um bis zu 45 Prozent reduziert, verfolgt primär drei Ziele. Zum einen spart sie Speicherplatz ein, denn große Help-Dateien können bis zu mehreren hundert Kbyte in Anspruch nehmen, und wenn die Göße einer solchen Datei im Hinblick auf die Speicherkapazität einer Festplatte auch kaum ins Gewicht fällt, so ist dieser Faktor im Rahmen der Distribution Ihrer Programme sehr wohl von Bedeutung. Wenn Sie Ihre Programme nämlich auf Diskette ausliefern, so kann leicht der Fall eintreten, daß eine große Help-Datei nicht mehr auf die Lieferdisketten paßt. Wenn Sie dann nur wegen dieser Help-Datei eine zusätzliche Diskette ausliefern müssen, kann das Ihre Produktionskosten unangenehm erhöhen.

Des weiteren - und dies ist der zweite Grund für die Kompression der Help-Datei - wirkt sich die Verkleinerung der Help-Datei auch positiv auf die Geschwindigkeit der Help-Engine aus. Hier muß berücksichtigt werden, daß die Help-Datei nicht am Programmanfang komplett in den Speicher geladen wird, sondern die einzelnen Hilfetexte nur bei Bedarf aus dieser Datei extrahiert und geladen werden. Da der Zugriff auf Massenspeicher im Verhältnis zur internen Verarbeitungsgeschwindigkeit des Prozessors viel Zeit in Anspruch nimmt (und die Massenspeicher im Vergleich zu immer schnelleren Prozessoren und höheren Taktraten immer langsamer werden) kann dank der Kompression beim Einlesen des Hilfetexts Zeit gespart werden.

Dieses Argument führt auch zu einem weiteren Pluspunkt für die Kompression der Help-Datei. Um nämlich nicht zu sehr von langsamen Massenspeichern gebremst zu werden, rüsten mittlerweile viele PC-Anwender ihre Rechner mit intelligenten Cache-Programmen aus, die sich die jeweils zuletzt gelesenen Daten merken und sie bei erneutem Abruf direkt und ohne Zugriff auf den Massenspeicher an den Aufrufer übergeben. Je kleiner die einzelnen Hilfetexte sind, desto mehr von ihnen kann das Cache-Programm im Speicher halten und desto seltener muß auf das Speichermedium zugegriffen werden.

Auf welchen Algorithmen diese Kompression beruht, erfahren die technisch Interessierten unter Ihnen im letzten Abschnitt dieses Artikels, doch lassen Sie uns zuvor den Aufbau der Help-Datei, wie sie der Help-Compiler erwartet, untersuchen.

Aufbau der Help-Datei

Wie oben bereits erwähnt, erwartet der Help-Compiler als Eingabedatei eine unformatierte ASCII-Datei, die die einzelnen Hilfetexte enthält. Damit der Help-Compiler die einzelnen Hilfetexte auseinanderhalten kann, muß jedem Hilfetext eine Zeile vorangehen, die mit einem Doppelpunkt beginnt. Diese Zeile dient allerdings nicht nur zur Separierung der verschiedenen Hilfetexte, sondern gibt auch gleichzitig die Nummer des Hilfetexts an, die als Dezimalzahl hinter dem Doppelpunkt folgen muß. In einem der obigen Abschnitte wurde bereits darauf hingewiesen, daß die einzelnen Hilfetexte Nummern zwischen 1 und 32.000 tragen können.

Zusätzlich sei hier noch vermerkt, daß die einzelnen Hilfetexte innerhalb der Help-Datei weder in irgendeiner Reihenfolge angegeben werden müssen, noch kontinuierliche Nummern tragen müssen. Bis auf den Hilfetext mit der Nummer 1, der grundsätzlich Informationen über die Hilfe-Funktion und ihre Bedienung enthalten muß, unterliegt auch der Inhalt der einzelnen Hilfetexte keinerlei Restriktionen. (Sexistische Anspielungen, Lästereien über Microsoft-Produkte und Witze über die Regierung der Bundesrepublik Deutschland werden vom Help-Compiler allerdings ersatzlos gestrichen und an die Bundesbehörde für die Verbreitung jugendgefährdender Schriften weitergeleitet.)

Was die Größe der einzelnen Hilfetexte anbelangt, so gibt die Help-Engine eine maximale Zeilenlänge von 255 Zeilen vor. Dies darf jedoch nicht als Aufforderung zum Schreiben endlos langer Hilfetexte mißverstanden werden. Vielmehr sollte ein Hilfetext kurz und prägnant sein und

eine Länge von 50 Zeilen daher nur in Ausnahmefällen überschritten werden. Die maximale Breite eines Hilfetexts kann durch die Konstante `MAX_Z_LEN` (innerhalb der Datei `HLPI.H`) eingestellt werden. Derzeit beträgt sie 180 Zeichen.

Neben den Zeilen, die durch einen Doppelpunkt eingeleitet werden, mißt der Help-Compiler auch den Zeilen eine besondere Bedeutung bei, die mit einem Semikolon beginnen. Sie betrachtet er als Kommentarzeilen, die er unbeachtet läßt und nicht in den jeweiligen Hilfetext übernimmt. Darüber hinaus kennt der Help-Compiler einige Befehle, die an beliebiger Stelle in einen Hilfetext eingefügt werden können und grundsätzlich mit einem Backslash (`'\'`) beginnen (*Tabelle 1*). Damit ungeachtet dessen auf den Backslash als normales Zeichen nicht verzichtet werden muß, steht mit dem `\\`-Befehl eine Möglichkeit für die Eingabe normaler Backslashs zur Verfügung.

Der Befehl `\xhh` gibt Ihnen die Möglichkeit, auch ASCII-Zeichen in einen Hilfetext aufzunehmen, deren Eingabe Ihr Editor oder Textverarbeitungsprogramm nicht erlaubt (in der Regel die Zeichen mit den ASCII-Codes kleiner 32). Die Buchstabenkombination `hh` innerhalb des Befehls muß dabei den hexadezimalen ASCII-Code des Zeichens widerspiegeln. Bei der Anzeige des Hilfetexts auf dem Bildschirm erscheint dann nicht dieser Befehl, sondern das durch ihn repräsentierte ASCII-Zeichen.

Möchten Sie bestimmte Zeichen innerhalb eines Hilfetexts besonders hervorheben, können Sie sich dazu des `\u`-Befehls bedienen. Stellen Sie diesen Befehl dem ersten hervorzuhebenden Zeichen voran und stellen sie dem letzten dieser Zeichen den Befehl `\n` nach, damit die folgenden Zeichen wieder normal dargestellt werden. Das `u` steht dabei übrigens für »unterstreichen«, denn auf einem monochromen Bildschirm werden diese Zeichen unterstrichen (während sie auf einem Color-Bildschirm in einer besonderen Farbe dargestellt werden).

Eine wichtige Bedeutung kommt auch dem `\v`-Befehl zu, der einen sogenannten Verweis markiert und einem Kapitelverweis wie »siehe Seite ...« in einem Buch gleichkommt. Im Gegensatz zu einem Buch enthält ein elektronischer Verweis jedoch die Seite, bzw. den Hilfetext, auf den er verweist, ohne daß er auf dem Bildschirm für den Anwender sichtbar würde. Wählt der Anwender jedoch einen Verweis an, holt die Help-Engine automatisch den Hilfetext, auf den verwiesen wurde, auf den Bildschirm.

Da Verweise auf dem Bildschirm farblich hervorgehoben werden, muß der Help-Compiler auch das Ende eines Verweises erkennen. Dem letzten Zeichen eines Verweises muß deshalb wiederum der `\v`-Befehl folgen. Diesmal jedoch muß diesem Befehl die Nummer des Hilfetexts (in dezimaler Notation), abgeschlossen durch einen Punkt, folgen. Beachten Sie dabei bitte, daß sich ein Verweis über maximal eine Zeile erstrecken darf, sich die beiden `\v`-Befehle also in ein und derselben Zeile befinden müssen. Allerdings spricht nichts dagegen, mehrere Verweise in einer Zeile anzubringen, denn der Help-Compiler erlaubt bis zu 255 Verweise pro Hilfetext, die beliebig über den Text verstreut werden können. Wie ein solcher Verweis innerhalb einer Help-Datei markiert werden kann, zeigt die folgende Beispielzeile, in der ein Verweis auf den Hilfetext mit der Nummer 73 aufgeführt wird:

```
Text ... \vdies ist ein Verweis\v73. ... Text
```

Dieser Art von Verweisen kommt neben der Verknüpfung einzelner Hilfetexte auch die Aufgabe zu, dem Anwender innerhalb eines kontextsensitiven Hilfe-Systems den Zugriff auf allgemeine Informationen zu ermöglichen. Viele PC-Applikationen fügen deshalb an das Ende jedes Hilfetexts einen Verweis auf einen sogenannten »Index« an. Vergleichbar mit dem Inhalts- oder Stichwortverzeichnis eines Buches findet der Anwender hier ausschließlich Verweise auf allgemeine Themen, etwa über die Bedienung des Programms, die Bedeutung der einzelnen Befehle, wichtige Tastenkombinationen usw.

Kann der Help-Compiler Ihre Help-Datei einwandfrei kompilieren, erstellt er eine kompilierte Help-Datei unter dem Namen der Eingabedatei, allerdings mit der Erweiterung ».HLP«. Dieser Vorgang muß nur einmal vollzogen und erst wiederholt werden, wenn Änderungen an der Quelldatei vorgenommen werden.

Beachten Sie bei der Umwandlung des Help-Compilers aus der Datei »HLPCOMP.C« in ein ausführbares EXE-Programm bitte, daß der Compiler im Gegensatz zu den Modulen unserer SAA-Reihe nicht unter jedem beliebigen Speichermodell kompiliert werden kann. Vielmehr wurde der Help-Compiler speziell für die Kompilierung unter dem Speichermodell `COMPACT` entwickelt und darf nicht unter einem anderen Speichermodell kompiliert werden.

Das SAA-Help-Modul

Das Gegenstück zum Help-Compiler bildet das Help-Modul `HLP.C`, welches das eigentliche Kernstück der SAA-Help-Engine darstellt. Es kann die durch den Help-Compiler erstellten Help-Dateien lesen, einzelne Hilfetexte daraus extrahieren und sie innerhalb eines Help-Fensters auf dem Bildschirm darstellen. Obwohl dieses Modul sehr groß ist, sind es nur drei Funktionen und eine Reihe kleinerer Makros, die von einer SAA-Applikation zum Zugriff auf die Help-Engine benötigt werden.

Um auf diese Funktionen und Makros zurückgreifen zu können, muß die SAA-Applikation neben den Include-Dateien der anderen SAA-Module auch die Include-Datei des Help-Moduls, die Datei `HLP.H`, einbinden.

Unter den verschiedenen Funktionen des Help-Moduls steht zunächst die Funktion `HlpInit()` im Vordergrund, die analog zu den Funktionen `VioInit()` (Video-Modul)

Befehl	Bedeutung
\xhh	Zeichen mit Hex-Code hh in Hilfetext aufnehmen
\u	nachfolgende Zeichen besonders hervorheben
\n	nachfolgende Zeichen wieder normal darstellen
\\	normaler Backslash
\v	Verweisanfang
\vD.	Verweisende. Verweis zeigt auf Hilfetext mit der Nummer D.

Tabelle 1: *Befehle, die vom Help-Compiler in besonderer Art und Weise interpretiert werden.*

mono	color	Makro	Bedeutung
MON_N	COL_N	HlpSCHText	normaler Hilfetext
MON_U	COL_U	HlpSCUnd	hervorgehobener Hilfetext
MON_V	COL_V	HlpSCVer	Verweis
MON_A	COL_A	HlpSCVerA	aktueller Verweis
MON_FR	COL_FR	HlpSCFrame	Rahmen um Help-Fenster
MON_BU	COL_BU	HlpSCBut	Felder in vorletzter Zeile
MON_SL	COL_SL	HlpSCSlid	verti. & horiz. Slider-Box
MON_PF	COL_PF	HlpSCPfeil	Pfeil an Slider-Boxen

Tabelle 2: *Konstanten, die die Farben innerhalb eines Help-Fensters bestimmen, und Makros, mit deren Hilfe diese Farben auch noch während des Programmablaufs verändert werden können.*

und `KbmInit()` (Tastatur- und Mausmodul) zur Initialisierung des Help-Moduls aufgerufen werden muß. Bei diesem Aufruf muß ihr der Name der zu bearbeitenden Help-Datei übergeben werden. Aus dieser Datei liest sie die Dekodiertabelle sowie einige Statusinformationen, wie die Anzahl der Hilfetexte, ihre Lage innerhalb der Help-Datei etc., ein. Anhand eines bestimmten Codes überprüft sie dabei die Konsistenz der Datei, denn bereits minimale Fehler - vor allem innerhalb der Dekodiertabelle - können beim Zugriff auf die einzelnen Hilfetexte zu einem Systemabsturz führen. Tritt ein solcher Fehler auf oder ist der Zugriff auf die Help-Datei nicht möglich, liefert die `HlpInit()`-Funktion ihrem Aufruf den Wert `FALSE` zurück. Er sollte die SAA-Applikation dazu veranlassen, das Programm entweder mit einer Fehlermeldung zu beenden oder zumindest im weiteren Verlauf des Programms auf den Aufruf der anderen Help-Funktionen zu verzichten. Werden Dekodiertabelle und Statusinformationen einwandfrei gelesen, liefert die Funktion den Wert `TRUE` zurück und dem Zugriff auf die einzelnen Hilfetexte steht nun nichts mehr im Wege.

Dieser erfolgt, wie bereits beschrieben, innerhalb einer speziellen Help-Funktion, die oben mit dem Namen `HelpFkt()` bezeichnet wurde. Sie kann sich zur Anzeige eines Hilfetexts der Funktion `HlpShow()` aus dem Help-Modul bedienen, der die Nummer des anzuzeigenden Hilfetexts übergeben werden muß. Sie öffnet zunächst das Help-Fenster, das alle andern Fenster überlagert. Dessen Größe und Position auf dem Bildschirm werden durch vier Konstanten bestimmt, die innerhalb der Include-Datei `HLP.H` definiert werden. `HF_X0` und `HF_Y0` spiegeln dabei die Koordinate der oberen linken Ecke des Help-Fensters wieder, während `HF_X` und `HF_Y` die Länge bzw. Breite des Help-Fensters angeben. Beachten Sie bitte, daß es sich bei diesen Konstanten nur um Initialwerte handelt, die jederzeit durch den Aufruf der Funktion `HlpSetHF()` überschrieben werden können. Solange aber kein Aufruf dieser Funktion erfolgt, erscheint das Help-Fenster an der durch diese Konstanten beschriebenen Position.

Während die maximale Größe dieses Fensters nur durch die Größe des Bildschirms eingeschränkt wird (vergessen Sie bei Ihren Berechnungen bitte nicht den Schatten unter dem Fenster), setzen andere Kriterien der Verkleinerung des Fensters gewisse Grenzen. So zeigt *Bild 1* z.B. die drei Felder in der vorletzten Zeile des Help-Fensters. Da sie grundsätzlich in dieser Zeile Platz finden sollen, muß das Help-Fenster mindestens 38 Zeichen breit sein. Auch eine Mindestlänge von 5 Zeilen darf nicht unterschritten werden, da allein 4 Zeilen für den Rahmen, die Slider etc. benötigt werden.

Neben dieser »physikalischen« Grenze stellt sich jedoch die Frage, ob es Sinn macht, ein Help-Fenster aufzubauen, in dem jeweils nur eine Zeile aus dem Hilfetext sichtbar wird. Sicherlich nicht, und so sollte ein Help-Fenster mindestens 14 Zeilen umfassen, damit gleichzeitig jeweils 10 Zeilen aus dem Hilfetext angezeigt werden können.

Trotz der Möglichkeit, die Größe des Help-Fensters und seine Position noch während des Programmablaufs zu verändern, empfehle ich Ihnen, zumindest die Größe dieses Fensters bereits bei der Programmerstellung definitiv festzulegen und diese Daten auch bei der Erstellung der Hilfetexte zu beachten. Gerade in Bezug auf die Breite des Help-Fensters kann nämlich sonst der sehr unangenehme Effekt eintreten, daß nicht der gesamte Teil der verschiedenen Zeilen gleichzeitig sichtbar ist. Zwar erlaubt das Help-Modul auch das seitliche Scrolling des Hilfetexts innerhalb des Help-Fensters, doch ist dies dem Durchblättern des Textes nicht gerade förderlich.

Neben den oben beschriebenen Konstanten bestimmten noch eine ganze Reihe weiterer Konstanten das Erscheinungsbild des Help-Fensters. Sie legen die Farbe der einzelnen Komponenten innerhalb eines Help-Fensters fest, wobei jeweils ein Satz dieser Konstanten für die Ausgabe auf einem Monochrom- bzw. einem Color-Monitor existiert. Auch diese Konstanten dienen wiederum nur als Initialwerte. Stellt die Funktion `HlpInit()` bei ihrem Aufruf fest, daß die Applikation im Monochrom-Modus betrieben wird, lädt sie die Monochrom-Konstanten in die verschiedenen Variablen, während beim Anschluß eines Color-Monitors die verschiedenen Color-Konstanten zum Einsatz kommen. Darüber hinaus können die Farben einzelner Komponenten auch noch während des Programmablaufs mit Hilfe einer ganzen Reihe von Makros manipuliert wer-

den, denen jeweils die neue Farbe für das Objekt übergeben werden muß. Gültig werden diese Veränderungen dann jeweils beim nächsten Aufruf der Funktion `HlpShow( )`.

Zur besseren Übersicht zeigt Ihnen die *Tabelle 2* die Bedeutung der verschiedenen Konstanten und den Namen des jeweils zugehörigen Makros.

Durchblättern eines Hilfetexts

Da ein Hilfetext nur in den seltensten Fällen komplett innerhalb des Help-Fensters dargestellt werden kann, stellt das Help-Modul dem Anwender alle Möglichkeiten bereit, beliebige Teile des Hilfetexts in den sichtbaren Teil des Help-Fensters zu holen. Von Seiten der Tastatur her wird dies durch die Cursor-Tasten unterstützt. Beachten Sie aber bitte, daß die Betätigung dieser Tasten nur Wirkung zeigt, wenn der Hilfetext tatsächlich über den sichtbaren Teil des Help-Fensters hinausragt. Wenn ein Hilfetext zwar länger, nicht aber breiter als das Help-Fenster ist, dann kann der Anwender zwar mit Hilfe der Tasten [↑], [↓], [Home], [End], [PgUp] und [PgDn] durch den Text blättern, doch bleibt die Betätigung der Tasten [→] und [←] erfolglos.

Sofern sich innerhalb des angezeigten Ausschnitts des Hilfetexts mindestens ein Verweis befindet, gibt es einen »aktuellen« Verweis, der farblich besonders hervorgehoben wird. Betätigt der Anwender die Taste [↵], wird der Hilfetext, auf den dieser Verweis zeigt, in des Help-Fenster geholt. Bei der Anzeige mehrerer Verweise hat der Anwender die Möglichkeit, den aktuellen Verweis mit Hilfe der Tasten [Tab] und [Shift][Tab] zu markieren.

Ein ganz besonderer Hilfetext, nämlich der Hilfetext Nummer 1 mit Informationen über die Arbeit mit der Hilfe-Funktion, wird vom Help-Modul immer dann in das Help-Fenster geladen, wenn der Anwender während der Anzeige eines Hilfetexts nochmals die Taste [F1] betätigt. Will der Anwender aus diesem Hilfetext in den zuvor angezeigten Text zurückkehren, kann er sich eine Funktion zu Nutze machen, die ihm auch in Verbindung mit jedem anderen Hilfetext zur Verfügung steht: Bei Betätigung der Tastenkombination [Alt][F1] holt das Help-Modul den jeweils zuvor betrachteten Hilfetext wieder in das Help-Fenster. Um wieviele Hilfetexte dabei zurückgeblättert werden kann, bestimmt die Konstante `ANZ_MERKE`, die innerhalb des Help-Moduls `HLP.C` definiert wird. Sie bestimmt die Größe des Help-Stacks, auf dem jeweils die Nummern der letzten n Hilfetexte gespeichert werden. Ist dieser Stack leer, wird pauschal der Hilfetext mit der Nummer `STACK_EMPTY` eingeladen.

Die Wiederaufnahme der Applikation erfolgt durch Betätigung der Taste [Esc], durch die das Help-Fenster geschlossen und wieder vom Bildschirm entfernt wird.

Alle Funktionen, die über die Tastatur erreicht werden können, können auch mit Hilfe einer Maus ausgelöst werden. Zum Durchblättern des Hilfetexts dienen dabei die beiden Slider-Boxen, die ihren Namen dem Slider, einem Schlitten, verdanken, der auf ihnen hin- und herbewegt werden kann und die Position des sichtbaren Textausschnitts innerhalb des gesamten Hilfetexts widerspiegelt. Wird der Maus-Cursor auf einen der beiden Slider bewegt und dann der linke Mausknopf niedergedrückt, bewegt sich der Slider (das durchsichtige Kästchen) auf den Maus-Cursor zu, wobei gleichzeitig auch der sichtbare Ausschnitt des Hilfetexts verschoben wird. Bleibt der linke Mausknopf weiterhin niedergedrückt, wiederholt sich dieser Vorgang, bis der Slider den Maus-Cursor eingeholt hat.

Wird der Maus-Cursor jedoch direkt auf einen der beiden Slider geführt und dann der linke Mausknopf niedergedrückt, so führt das Help-Modul den Slider hinter dem Maus-Cursor her, bis der linke Mausknopf losgelassen wird. Der sichtbare Textausschnitt wird daraufhin so verschoben, daß die Slider-Position die Lage des Textausschnitts widergibt.

Ähnlich den beiden Slidern führen auch die Pfeile am Rand der Slider zu einer Verschiebung des Textausschnitts, sobald der Maus-Cursor auf sie bewegt und der linke Mausknopf niedergedrückt wird. Der sichtbare Textausschnitt wird dabei jeweils um eine Spalte bzw. Zeile in die Richtung des Pfeils verschoben.

Beachten Sie aber bitte, daß auch in Verbindung mit der Maus eine Verschiebung des Textausschnitts nur möglich ist, wenn der angezeigte Hilfetext über das Help-Fenster hinausreicht.

Die Auswahl eines Verweises mit der Maus erfolgt einfach, indem der Maus-Cursor auf den gewünschten Verweis bewegt und dann der linke Mausknopf niedergedrückt wird. Auf die gleiche Art und Weise können auch die Befehle ausgelöst werden, die in der vorletzten Zeile des Help-Fensters aufgeführt sind.

Das Demo-Programm

Auch diese Folge unserer SAA-Serie soll nicht enden, ohne Ihnen die Fähigkeiten des vorgestellten Moduls an einem Beispiel zu demonstrieren. Im Rahmen dieser Folge ist jedoch weniger das Demo-Programm selbst, sondern vielmehr die zugehörigen Daten, sprich die Help-Datei interessant. Diese Demo-Datei, mit dem Namen `HLPDEMO.T` finden Sie auf den folgenden Seiten abgedruckt. Um sie in das `HLP`-Format umzuwandeln, müssen Sie zunächst den Help-Compiler `HLPCOMP` in ein ausführbares EXE-Programm umwandeln (siehe dazu Informationen im Kopf des Programm-Listings) und ihn dann mit dem Namen der Demo-Datei aufrufen.

```
hlpcomp hlpdemo.t
```

Die während der Ausführung erstellte Datei `HLPDEMO.HLP` dient dann als Help-Datei für das Demoprogamm `HLPDEMO.C`.

Datenkompression nach Huffman

Wie bereits am Anfang dieses Artikel versprochen, möchte ich allen Leser, die sich für die Datenkompression innerhalb der Help-Engine interessieren, noch kurz in die Theorie des hier zugrundeliegenen Kompressions-Algorithmus einführen. Er läßt sich auf eine Veröffentlichung des französischen Mathematikers D.A.Huffman aus dem Jahre 1952 zurückführen und stellt neben dem sogenannten »Ziff-Lempel-Welsh-Coding« das derzeit populärste Kompressionsschema dar. Huffman bedient sich einer Idee, die bereits den legendären Samuel Morse bei der Entwicklung seines gleichnamigen Morse-Alphabets beeinflußt hat.

Morse verzichtete auf die Informationskodierung in konstanten Datenpaketen, wie sie heute etwa im 8-Bit-ASCII-Zeichensatz zum Ausdruck kommt, und wandte sich variablen Codelängen zu, die den Häufigkeiten der einzelnen Daten Rechnung trugen. Morse, dem es um die Nachrichtenübertragung mit Hilfe eines Telegraphennetzes ging, untersuchte dazu die Häufigkeiten seiner Daten, also die Häufigkeiten der einzelnen Zeichen innerhalb der englischen Sprache und stellte dabei große Unterschiede fest. Die Bandbreite reichte von einer knapp 15 prozentigen Häufigkeit des Buchstabens e bis hinunter zu einem hundertstel Prozent für den Buchstaben x. Dies ausnutzend vergab Morse den häufigeren Zeichen kürzere Codes, als den weniger häufig auftretenden Zeichen und konnte dadurch eine wesentlich geringe Codelänge und gleichsam höhere Informationsdichte erzielen, als es mit einem herkömmlichen Code möglich wäre.

Nicht anders ging Huffman vor, nur daß er die Ideen Morses auf eine algorithmische Grundlage stellte und so zeigte, wie man beliebige Datenmengen mit Hilfe eines Huffman-Codes komprimieren und dadurch bis zu 50% und mehr Speicherplatz sparen kann. Als schwierigstes Problem galt es dabei, das Problem der Redundanz, also den Beginn verschiedener Datencodes mit identischen Codefolgen, zu verhindern. Huffman ermittelt dazu zunächst die Häufigkeiten der verschiedenen Daten (hier: der Zeichen innerhalb der Help-Datei) und sortiert die einzelnen Daten dann in absteigender Reihenfolge in eine Liste (in C: Vektor) ein.

Diese Liste bildet den Ausgangspunkt für die Erstellung eines binären Baums, aus dem später die Codefolgen der einzelnen Daten ermittelt werden können. Es werden dazu zunächst die letzten beiden Einträge innerhalb der Liste zu den Blättern eines Knotens zusammengefaßt und die Häufigkeit der durch sie repräsentierten Daten addiert. Je nach der daraus resultierende Summe wird der erstellte Knoten wieder in die Liste einsortiert, wobei er in der Regel nach oben wandert, da er nun mehr Zeichen vertritt, als beispielsweise der unmittelbar vorhergehende Listeneintrag.

Dieser Vorgang wird nun Schritt für Schritt wiederholt, wobei sich nach einiger Zeit bereits erstellte Knoten wieder am Ende der Liste finden und als die beiden Blätter eines Knotens zusammengeführt werden. Dabei bilden sich die ersten Teile des Baumes. Durch das Verbinden jeweils zweier Listeneinträge schrumpft die Liste mit jedem Durchlauf um einen Eintrag und enthält schließlich nur noch einen Eintrag, der alle Daten repräsentiert und den gewünschten Codebaum darstellt. Die Codes der einzelnen Daten können diesem Codebaum entnommen werden, inem dieser Baum rekursiv durchlaufen wird, bis alle Blätter erreicht wurden. An der Wurzel beginnt man dabei mit einem leeren Code und fügt für jede Rekursion nach links eine 1 an das Ende des Codes an, während bei einer Rekursion nach rechts eine 0 an den Code angehangen wird. (Man kann 0 und 1 auch vertauschen, da dies ledigich den Code negiert, nicht aber grundsätzlich verändert.)

Gelangt man dabei an ein Blatt, so repräsentiert der bis dato aufgereihte Code die Codefolge das zugehörigen Datums. Dabei kann man leicht eine Codetabelle erstellen, bei der das jeweilige Datum (in unserem Fall: der ASCII-Code des Zeichens) über eine Ordinal-Funktion als Index in diese Tabelle abgebildet wird. Das indizierte Element nimmt dabei jeweils die Codefolge des Datums auf.

Nach der Erstellung dieses Codebaums ist die Datenkompression nicht mehr schwer, denn anstatt des jeweiligen Datums muß nur noch die Codefolge des Datums aus der Codetabelle ermittelt und ausgegeben werden. Gerade bei der Ausgabe in eine Datei darf allerdings nicht vergessen werden, den Codebaum ebenfalls auszugeben, denn ohne ihn kann die Datei später nicht mehr richtig dekomprimiert werden.

Michael Tischer

```c
/*
[ ]------------------------------------------------------------[ ]
 |                                                               |
 |     Include-Datei     : HLPI.H                                |
 |     zur Einbindung    : der internen Deklarationen für        |
 |                         Programme HELPCOMP.C und HLP.C        |
 |     erstellt am       :  8.05.1989                            |
 |     letztes Update am: 13.05.1989                             |
 |                                                               |
 |     (Copyright)       : 1989 by MICHAEL TISCHER               |
 |                                                               |
[ ]------------------------------------------------------------[ ]
*/

/*-- Typedefs ----------------------------------------------------*/

#ifndef BYTE                          /* wir basteln uns ein Byte */
 typedef unsigned char BYTE;
#endif

#ifndef WORD
 typedef unsigned int WORD;
#endif

#ifndef BOOL                            /* wie BOOLEAN in Pascal */
 typedef BYTE BOOL;
#endif

typedef struct node   NODE;    /* Knoten oder Blatt im Baum */
typedef NODE *        KNOPER;          /* Pointer auf Knoten */
typedef BYTE *        BPTR;              /* Pointer auf Byte */
typedef struct hcdes  HCD;      /* Position des Help-Texts */
typedef HCD *         HCDP;

/*-- Konstanten --------------------------------------------------*/

#define RG    register

#ifndef TRUE
  #define TRUE  ( 1 == 1 )
  #define FALSE ( 1 == 0 )
#endif

#define KNOTEN 0
#define BLATT  1
#define NIL    ((void *) 0) /* kein Nachfolger in der Liste */

#define ID_CODE 0x544d           /* identifiziert Help-Datei */

#define HILFE_TEXT 1     /* Hilfs-Text mit Infos über Hilfe */

/*-- Tokens innerhalb einer kompilierten Help-Seite --------*/

#define TOK_UNTER   0   /* Zeichen werden unterstrichen     */
#define TOK_NORMAL  1   /* Zeichen werden normal dargstellt */
#define TOK_VERWEIS 2   /* Anfang oder Ende eines Verweises */
#define TOK_ZENDE   3   /* Zeilenende                       */
#define TOK_CODE    4   /* Zeichencode im nächsten Byte     */
                        /* TOK_CODE muß immer das Token mit */
                        /* dem größten Code sein!           */

#define START_STR "SAA Help-Datei!\x1a"
#define EXT        ".HLP"   /* Erweiterung kompilierte Datei */

#define MAX_Z_LEN  180              /* maximale Zeilenlänge */
#define MAX_G_LEN  12000  /*maximale Gesamtlänge einer Seite*/

/*-- Strukturen und Unions ---------------------------------*/

struct node         /* ein Knoten oder Blatt im binären Baum */
 {
  BYTE   typ,                          /* Blatt oder Knoten */
         ascii;      /* Codes des ASCII-Zeichen wenn Blatt */
  KNOPER links,            /* Pointer auf linken Nachfolger */
         rechts;          /* Pointer auf rechten Nachfolger */
  WORD   anzahl;                /* Anzahl der Zeichen, die  */
 };                             /* dieser Node repräsentiert */

struct hpinfo    /* geht jedem kompilierten Help-Text voran */
 {
  BYTE verweise,                          /* Anzahl Verweise */
       spalten,                     /* größte Spaltenbreite */
       zeilen;                     /* Anzahl der Textzeilen */
 };

struct header                         /* Kopf einer Help-Datei */
 {
  char str[ sizeof START_STR ];          /* Anti-Type-String */
  WORD code;                      /* ID-Code, muß 544Dh sein */
  int  anzahl;   /* Anzahl der Help-Seiten in dieser Datei */
 };

struct hcdes             /* beschreibt kompilierten Help-Text */
 {
  int  hnr,                            /* Nummer des Help-Texts */
       len;                             /* Länge dekomprimiert */
  long fpos;              /* Position innerhalb der Help-Datei */
 };
```

Listing 1: HLPI.H

```c
/************************************************************/
/*                     H L P C O M P . C                    */
/*----------------------------------------------------------*/
/*     Aufgabe        : Komiliert und Help-Datei und bringt */
/*                      sie in das von der Help-Engine (Mo- */
/*                      dul HLP.C) erwartete Format         */
/*----------------------------------------------------------*/
/*     Autor          : MICHAEL TISCHER                     */
/*     entwickelt am  :  9.05.1989                          */
/*     letztes Update : 13.05.1989                          */
/*----------------------------------------------------------*/
/*     Erstellung     : CL /AC HLPCOMP.C                    */
/*     Aufruf         : HLPCOMP [ Help-Datei ]              */
/*                      die erstellte Help-Datei wird mit   */
/*                      der Erweiterung ".HLP" versehen.    */
/************************************************************/

/*== Include-Dateien einbinden =============================*/

#include <dos.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <setjmp.h>
#include <string.h>
#include <malloc.h>
#include <ctype.h>
#include "hlpi.h"   /* Help-interne Deklarationen einbinden */

/*== Typedefs ==============================================*/

typedef struct codel  UBES;            /* eine Bit-Sequenz */
typedef struct hpdes  HPD; /* HText-Beschreiber im Speicher */
typedef HPD *         HPDP;

/*== Makros ================================================*/

/*-- die folgenden Makros dienen der Funktion COMPILE() ----*/
/*-- und der von ihr aufgerufenen Funktionen           ----*/

#define PUTCH(x) ( *bptr++ = x, ++anzb )
#define GETCH()  ( *zptr++ )

/*-- die folgenden Makros arbeiten mit der Funktion WRITE- -*/
/*-- BIT() zusammen.                                       -*/

#define WB_INIT( ptr )     ( wb_akt = wb_start = ptr )
#define WB_GET_LEN()       ( wb_akt - wb_start )
#define WB_ENDE()          ( write_bit( -1 ) )

/*== Strukturen und Unions =================================*/

struct codel  /* beschreibt die Bit-Sequenz für ein Zeichen */
 {
  BYTE len;                         /* Länge der Bit-Sequenz */
  char code[20];     /* Sequenz ist maximal 19 Zeichen lang */
 };

struct hpdes        /* geht dem Help-Text im Speicher voran */
 {
  struct hpdes * next;           /* Zeiger auf nächste Seite */
  int    len,                       /* dekomprimierte Länge */
         nummer;                  /* Nummer des Help-Textes */
  struct hpinfo i;
  BYTE   fb[1];                /* erstes Byte des Help-Texts */
 };

/*== Konstanten ============================================*/

#define KOMMENTAR ';'          /* leitet Kommentarzeile ein */
#define START     ':'        /* leitete neuen Help-Text ein */
#define LF        '\x0a'

/*-- Fehlercodes für die Fehlerbehandlungsroutine ----------*/

#define E_NOTEXIST  0 /* die Source-Datei existiert nicht   */
#define E_READ      1 /* Fehler bei Lesezugriff             */
#define E_NOMEM     2 /* nicht genügend Speicher            */
#define E_DEST      3 /* Öffnen der Ziel-Datei unmöglich    */
#define E_WRITE     4 /* Fehler bei Schreiben in Ziel-Datei */
#define E_CLOSE     5 /* Fehler bei Close auf Ziel-Datei    */
#define E_ZU_LANG   6 /* Zeile zu lang                      */
#define E_START_EX  7 /* Anfang einer Help-Seite erwartet   */
#define E_HELPNR    8 /* ungültige Help-Nummer              */
#define E_HLP_LEER  9 /* Help-Text ist leer                 */
#define E_2_LONG   10 /* Help-Text ist zu lang              */
#define E_UNBEF    11 /* unbekannter Befehlscode            */
#define E_VE_EXP   12 /* Verweis-Ende-Zeichen erwartet      */
#define E_P_EXP    13 /* Punkt erwartet                     */

#define PRINT_NUM   6     /* erster Fehler mit Zeilenangabe */

/*-- Zeichen-Codes, die bestimmte Befehle repräsentieren ---*/

#define CHR_INIT    '\\'/* leitet Befehl ein                 */
#define CHR_UNTER   'u' /* Zeichen werden unterstrichen     */
#define CHR_NORMAL  'n' /* Zeichen werden normal dargstellt */
#define CHR_VERWEIS 'v' /* Anfang oder Ende eines Verweises */
                        /* Interpretation ist kontext-sens. */
#define CHR_XY      'x' /* Zeichencode xy, wie in C ('\x1F')*/
#define CHR_VZENDE  '.' /* Ende der Zahl hinter einem Verw. */

/*== globale Variablen für Kompressionsteil ================*/

jmp_buf err_jmp;     /* nimmt Informationen für LONGJMP auf */
NODE    baum[256];                       /* der binäre Baum */
UBES    codetab[256];    /* nimmt Codierung der Zeichen auf */
```

Listing 2: HLPCOMP.C

```
/*== globale Variablen für Kompilationsteil ================*/

HPDP  start_hpd;                    /* Zeiger auf ersten HPD */
long  zeile = 1;        /* aktuelle Zeile in der Help-Datei */
FILE * fps;    /* Pointer auf Stream-Info über Source-Datei */
int   anzhelp;                   /* Anzahl der Help-Seiten */

/*-- die folgenden Variablen dienen nur der Funktion COM- --*/
/*-- PILE() und der von ihr aufgerufenen Funtkionen       --*/

char * zptr;              /* Zeiger auf den Anfang einer Zeile */
int   len,        /* Anzahl sichtbarer Zeichen in der Zeile */
      anzb;          /* Anzahl Zeichen in einer Help-Seite */
BPTR  bptr;           /* Zeiger in kompilierten Help-Text */

/*-- die folgenden Variablen dienen nur der Funktion  -----*/
/*-- WRITE_BIT und den Makros WB_INIT sowie WB_GET_LEN -----*/

BPTR wb_start,   /* Zeiger auf Anfang des WB-Ausgabepuffers */
     wb_akt;               /* Zeiger auf aktuelle Position */

/*== Kompilations-Teil =====================================*/

/************************************************************
*  Funktion          : A L L O C                             *
**---------------------------------------------------------**
*  Aufgabe           : Allokiert maximal jeweils 64 KByte    *
*                      auf dem Heap                          *
*  Eingabe-Parameter: LEN : Größe des zu allokierenden Spei- *
*                           chers in Byte.                   *
*  Return-Wert       : Zeiger auf den allokierten Speicher   *
*  Info              : Steht nicht genügend Speicher zur Ver- *
*                      fügung wird die Fehlerbehandlungs-Rou- *
*                      tine aufgerufen.                      *
************************************************************/

void * alloc( unsigned len )
{
 RG void * adr;       /* Zeiger auf den allokierten Speicher */

 if ( ( adr = malloc( len ) ) == NULL )
  longjmp( err_jmp, E_NOMEM );          /* zu wenig Speicher! */
 return adr;         /* alles o.k., Speicher wurde allokiert */
}

/************************************************************
*  Funktion          : G E T _ C H A R                       *
**---------------------------------------------------------**
*  Aufgabe           : Holt das nächste Zeichen aus der Help- *
*                      Datei.                                *
*  Eingabe-Parameter: keiner                                 *
*  Return-Wert       : das gelesense Zeichen oder EOF, wenn  *
*                      das Ende der Datei erreicht wurde.    *
************************************************************/

int get_char( void )
{
 RG int zeichen;                    /* das eingelesene Zeichen */

 if ( ( zeichen = fgetc( fps ) ) == EOF )
  {                            /* EOF erhalten, Lesefehler? */
   if ( ferror( fps ) )
    longjmp( err_jmp, E_READ );                         /* Ja! */
  }
 return zeichen;                     /* Zeichen zurückliefern */
}

/************************************************************
*  Funktion          : G E T _ L I N E                       *
**---------------------------------------------------------**
*  Aufgabe           : Holt die nächste Zeile aus der Help-  *
*                      Datei.                                *
*  Eingabe-Parameter: keiner                                 *
*  Return-Wert       : Zeiger auf einen Puffer mit der Zeile. *
*  Info              : - Kommentarzeilen werden überlesen und *
*                        das LF am Ende einer Zeile grundsätz-*
*                        lich durch '\0' ersetzt.            *
*                      - Konnte keine Zeile mehr gelesen wer- *
*                        den, weil das Dateiende bereits er-  *
*                        reicht war, wird NULL zurückgeliefert*
************************************************************/
char * get_line( void )
{
 static char zbuf[ MAX_Z_LEN + 1 ];  /* nimmt die Zeile auf */
 static BOOL ende = FALSE;

 RG int zeichen,                   /* das eingelesene Zeichen */
        pos;                       /* Nummer des Zeichens */
 do                                         /* Leseschleife */
  {
   if ( ende )                        /* EOF bereits gelesen? */
    return NULL;                   /* Ja, NULL zurückliefern */
   pos = 0;                    /* mit erstem Zeichen beginnen */
   do                        /* eine kompeltte Zeile einlesen */
    {
     ende = ( ( zeichen = get_char() ) == EOF );
     if ( ende  ||  zeichen == LF )
      zeichen = '\0';                            /* Zeilenende */
     zbuf[ pos++ ] = zeichen;    /* Zeichen in Puffer holen */
     if ( pos == MAX_Z_LEN + 1 )            /* Zeile zu lang? */
      longjmp( err_jmp, E_ZU_LANG );                    /* Ja! */
    }
   while ( zeichen );          /* wiederholen, bis Zeilenende */
   if ( ( ++zeile & 15 ) == 0 )   /* Zeilennummer ausgeben? */
    printf("\b\b\b\b\b%5ld", zeile );                    /* Ja */
  }
 while ( zbuf[0] == KOMMENTAR );
 return zbuf;            /* Zeiger auf die Zeile zurückliefern */
}
```

Listing 2: *(Fortsetzung)*

```
/************************************************************
*  Funktion          : G E T _ H E X                         *
**---------------------------------------------------------**
*  Aufgabe           : Liest eine zweiziffrige Hexadezimal-  *
*                      Zahl aus einem String und wandelt sie *
*                      nach ASCII um.                        *
*  Eingabe-Parameter: keine                                  *
*  Return-Wert       : Dezimal-Wert der Hex-Zahl             *
*  Info              : - Wird keine korrekte Hexadezimalzahl *
*                        angetroffen, wird die Fehlerbehand- *
*                        lungsroutine aufgerufen.            *
************************************************************/
BYTE get_hex( void )
{
 RG BYTE ziffer1,           /* Wertigkeit der beiden Ziffern */
         ziffer2;

 if ( !isxdigit( *zptr )  ||  !isxdigit( *(zptr+1) ) )
  longjmp( err_jmp, E_UNBEF );

 /*-- die erste Hexziffer nach Dezimal wandeln -------------*/
 if ( isalpha( *zptr ) )                      /* Buchstabe? */
  ziffer1 = toupper( *zptr ) - 'A' + 10;             /* Ja */
 else                                     /* Nein, Ziffer */
  ziffer1 = *zptr & 15;

 /*-- die zweite Hexziffer nach Dezimal wandeln ------------*/
 if ( isalpha( *++zptr ) )                    /* Buchstabe? */
  ziffer2 = toupper( *zptr ) - 'A' + 10;             /* Ja */
 else                                     /* Nein, Ziffer */
  ziffer2 = *zptr & 15;

 ++zptr;           /* Zeiger hinter zweite Hexziffer setzen */
 return ( ziffer1 << 4 ) + ziffer2;      /* Hex-Wert bilden */
}

/************************************************************
*  Funktion          : V E R W E I S                         *
**---------------------------------------------------------**
*  Aufgabe           : Wird aufgerufen, nachdem COMPILE()    *
*                      auf ein einleitendes Verweis-Zeichen  *
*                      gestoßen ist.                         *
*  Eingabe-Parameter: keine                                  *
*  Return-Wert       : keiner                                *
*  Info              : - Wird kein korrekter Verweis entdeckt,*
*                        wird die Fehlerbehandlungsroutnine  *
*                        aufgerufen.                         *
************************************************************/
void verweis( void )
{
 RG char zeichen,           /* jeweils bearbeitetes Zeichen */
         *merke;                   /* speichert Position */
 int     hnr;               /* Nummer des Verweis-Textes */
 BOOL    ende;

 ende = FALSE;                 /* jetzt geht's erst mal los */
 PUTCH( TOK_VERWEIS );          /* Verweis-Anfang merken */
 do                                         /* Leschleife */
  {
   switch ( zeichen = GETCH() )
    {
     case '\0' :                                 /* Zeilenende */
      longjmp( err_jmp, E_VE_EXP );                 /* Fehler */
      break;
     case CHR_INIT :
       switch( zeichen = GETCH() )     /* nächstes Z. holen */
        {
         case CHR_INIT      :         /* noch ein Backslash? */
           PUTCH( CHR_INIT );               /* Ja, übernehmen */
           ++len;                    /* Länge inkrementieren */
           break;

         case CHR_XY        :        /* Zeichen im Hex-Format */
           if ( ( zeichen = get_hex( ) ) <= TOK_CODE )
            PUTCH( TOK_CODE );
           PUTCH( zeichen );
           ++len;         /* ein Zeichen mehr in der Zeile */
           break;

         case CHR_VERWEIS :                    /* Verweis-Ende */
           PUTCH( TOK_VERWEIS );   /* Verweis-Ende anzeigen */
           merke = zptr;       /* aktuelle Position merken */
           while ( isdigit( GETCH() ) )         /* Zahl lesen */
             ;
           if ( *(zptr-1) != CHR_VZENDE )  /*Punkt gefunden?*/
            longjmp( err_jmp, E_P_EXP );             /* Fehler */

           /*-- die Zahl nach Integer wandeln und dann den -*/
           /*-- Code in den Puffer schreiben               -*/
           *(zptr-1) = '\0';           /* Zahl-String beenden */
           *( (int *) bptr ) = atoi( merke );
           bptr += 2;
           anzb += 2;
           ende = TRUE;                   /* Funktion beenden */
           break;

        default        :          /* jedes andere Zeichen */
           longjmp( err_jmp, E_VE_EXP );             /* Fehler */
        }
      break;

    default :                     /* jedes andere Zeichen */
      PUTCH( zeichen );               /* in Puffer bringen */
      ++len;                   /* Anzahl der Zeichen inkr. */
      break;
    }
  }
 while ( !ende );
}
```

Listing 2: *(Fortsetzung)*

```
/*************************************************************
*  Funktion          : C O M P I L E                        *
**---------------------------------------------------------**
*  Aufgabe           : Koordiniert die Kompilierung der ein-  *
*                      geladenen Help-Datei des angegebenen   *
*                      Speicherbereichs.                      *
*  Eingabe-Parameter: DATEI: Zeiger auf String mit dem Namen *
*                             der Datei                       *
*  Return-Wert       : keiner                                 *
*************************************************************/

void compile( char * datei )
{
 int     helpnr;         /* Nummer der aktuellen Help-Seite */
 BYTE    spalten,  /* maximale Spaltenbreite der Help-Seite */
         zeilen,          /* Anzahl Zeilen in der Help-Seite */
         zeichen,            /* aktuell bearbeitetes Zeichen */
         anzver;                     /* Anzahl der Verweise */
 HPDP    aktp;  /* Zeiger auf Puffer des vorherg. Help-Text */
 BOOL    token;            /* handelt es sich um ein TOKEN? */
 RG HPDP hptr;        /* Zeiger auf aktuellen Speicherblock */

 if ( ( fps = fopen( datei, "rt" ) ) == (FILE *) 0 )
  longjmp( err_jmp, E_NOTEXIST ); /* Datei existiert nicht! */

 anzhelp = 0;             /* noch keine Help-Seite bearbeitet */
 aktp = NIL;
 zptr = get_line();                      /* erste Zeile holen */
 do            /* eine Help-Seite einlesen und kompilieren */
  {
   if ( zptr )                 /* Ende der Datei erreicht? */
    {                                             /* Nein */
     if ( *zptr != START )               /* Start-Zeichen? */
      longjmp( err_jmp, E_START_EX );       /* Nein, Fehler */

     /*-- Nummer des Help-Textes ermitteln -----------------*/
     if ( ( helpnr = atoi( zptr+1 ) ) == 0 )
      longjmp( err_jmp, E_HELPNR );         /* ungültige Zahl */

     anzver = anzb = spalten = zeilen = 0;
     /*-- Puffer für Help-Text mit maximaler Länge allok. --*/
     hptr = (HPDP) alloc( MAX_G_LEN + sizeof( HPD ) );

     hptr->next = NIL;             /* noch keinen Nachfolger */
     bptr = hptr->fb;   /* Ptr auf erstes Byte im Help-Text */
     while( TRUE )  /* die einzelnen Help-Zeilen bearbeiten */
      {
       if ( (zptr = get_line())  &&  *zptr != START )
        {       /* weder letzte Zeile noch neuer Help-Text */
         ++zeilen;      /* Anzahl der Zeilen inkrementieren */
         len = 0;              /* Zeilenlänge initialisieren */

         while ( zeichen = GETCH() )  /*  Zeile durchlaufen */
          {
           if ( anzb > MAX_G_LEN )     /* zu viele Zeichen? */
            longjmp( err_jmp, E_2_LONG );                /* Ja */

           if ( token = ( zeichen == CHR_INIT ) )  /*Befehl?*/
            {      /* Ja, nachfolgendes Zeichen untersuchen */
             switch ( GETCH() )
              {
               case CHR_INIT    :           /* ein Backslash */
                 PUTCH( CHR_INIT );         /* ist kein Token */
                 break;

               case CHR_UNTER   : /* Zeichen unterstreichen */
                 PUTCH( TOK_UNTER );
                 break;

               case CHR_NORMAL  :            /* Zeichen normal */
                 PUTCH( TOK_NORMAL );        /* darstellen     */
                 break;

               case CHR_VERWEIS :       /* Einleitung Verweis */
                 verweis();              /* Verweis bearbeiten */
                 ++anzver;  /*Anzahl Verweise inkrementieren*/
                 break;

               case CHR_XY      :  /*Hex-Code eines Zeichens*/
                 if ( ( zeichen = get_hex( ) ) <= TOK_CODE )
                  PUTCH( TOK_CODE );
                 PUTCH( zeichen );
                 ++len;      /* Zeichen verlägert die Zeile */
                 break;

               default :   /* jedes andere Zeichen = Fehler */
                longjmp( err_jmp, E_UNBEF );
              }
            }
           else                         /* kein Befehlszeichen */
            {
             if ( zeichen <= TOK_CODE )
              PUTCH( TOK_CODE );              /* Escape-Zeichen */
             PUTCH( zeichen );             /* Zeichen in Puffer */
             ++len;               /* Zeilenlänge inkrementieren */
            }
          }
         PUTCH( TOK_ZENDE );                      /* Zeilenende */

         if ( len > spalten )       /* bisher längste Zeile? */
          spalten = len;                  /* Ja, Länge merken */
        }
       else             /* letzte Zeile oder neuer Help-Text */
        break;                            /* Schleife beenden */
      }
```

Listing 2: *(Fortsetzung)*

```
     /*-- die Bearbeitung eines Help-Texts wurde beendet ---*/

     if ( anzb )                   /* ist der Help-Text leer? */
      {              /* Nein, statistische Parameter laden */
       hptr->i.zeilen   = zeilen;
       hptr->i.spalten  = spalten;
       hptr->i.verweise = anzver;
       hptr->len        = anzb + sizeof hptr->i;
       hptr->nummer     = helpnr;
      }
     else                                        /* Ja, Fehler */
      longjmp( err_jmp, E_HLP_LEER );

     ++anzhelp;     /* Anzahl der Help-Texte inkrementieren */
     hptr = realloc( hptr, anzb - 1 + sizeof( HPD ) );
     if ( aktp == NIL )          /* war es erste Help-Seite? */
      start_hpd = hptr;                /* Ja, Pointer merken */
     else                                             /* Nein */
      aktp->next = hptr;        /* Verknüpfung zum Vorgänger */
     aktp = hptr;              /* HPTR wird der aktuelle HPD */
    }
  }
 while ( zptr );       /* Schleife bis Dateiende wiederholen */

 fclose( fps );                      /* Help-Datei schließen */
}

/*== Kompressions-Teil ====================================*/

/*************************************************************
*  Funktion          : O U T _ N A M E                       *
**---------------------------------------------------------**
*  Aufgabe           : Setzt den Namen der Ziel-Datei aus dem *
*                      Namen der Source-Datei zusammen.       *
*  Eingabe-Parameter: SOURCE: Pointer auf Namen der Source-  *
*                              Datei                          *
*  Return-Wert       : Pointer auf Namen der Ziel-Datei in    *
*                      einem statischen Puffer                *
*************************************************************/

char *out_name( char *source )
 {
  static char buf[65];   /* Puffer für Namen der Ziel-Datei */
  RG char * point;    /* Pointer auf letzten Punkt im Namen */

  strcpy(buf, source);  /* Namen in Ausgabe-Puffer kopieren */
  if ( point = strchr(buf, '.') )   /* gibt es einen Punkt? */
   strcpy(point, EXT); /* Ja, Erweiterung an Punkt anhängen */
  else                                                /* Nein */
   strcpy(buf + strlen(buf), EXT); /* an den Namen anhängen */
  return( buf );         /* Pointer auf Puffer zurückliefern */
 }

/*************************************************************
*  Funktion          : W R I T E _ B I T                     *
**---------------------------------------------------------**
*  Aufgabe           : Schreibt ein Bit an die aktuelle Posi- *
*                      tion innerhalb des Puffers, dessen     *
*                      Adresse über das Makro WB_INIT defi-   *
*                      niert wurde.                           *
*  Eingabe-Parameter: BIT: Wertigkeit des Bits.              *
*  Return-Wert       : keiner                                 *
*  Info              : - Bei einem Fehler wird die Fehlerbe-  *
*                        handlungs-Routine über LONGJMP ange- *
*                        sprungen.                            *
*                      - Ein Bit-Wert von -1 zeigt an, daß    *
*                        das aktuelle Byte ausgegeben werden  *
*                        soll, auch wenn es noch nicht voll   *
*                        ist.                                 *
*************************************************************/

void write_bit( int bit )
{
 static BYTE akt_bit = 0;                         /* Bit-Zähler */
 static BYTE akt_byte = 0;                /* das aktuelle Byte */

 if ( bit != -1 )                   /* nicht das letzte Bit? */
  {                                                   /* Nein */
   akt_byte |= bit;       /* das Bit in das Byte einblenden */
   if (akt_bit++ == 7)           /* ist das Byte jetzt voll? */
    {                                                   /* Ja */
     *wb_akt++ = akt_byte;         /* Byte in Puffer bringen */
     akt_bit = akt_byte = 0;   /* wieder von vorne anfangen */
    }
   else                     /* das Byte ist noch nicht voll */
    akt_byte <<= 1;  /* Inhalt des Bytes nach links shiften */
  }
 else                                         /* letztes Bit */
  if (akt_bit)       /* ist diese Byte schon "angebrochen"? */
   {                               /* Ja, Byte komplettieren */
    akt_byte <<= ( 7 - akt_bit );   /* Bits nach links sch. */
    *wb_akt++ = akt_byte;          /* Byte in Puffer bringen */
    akt_bit = akt_byte = 0;    /* wieder von vorne anfangen */
   }
}

/*************************************************************
*  Funktion          : W R I T E _ T R E E                   *
**---------------------------------------------------------**
*  Aufgabe           : Hangelt sich rekursiv durch den Baum,  *
*                      wobei für des Blatt eine 1 (gefolgt von*
*                      den 8 Bit des ASCII-Codes) und für je- *
*                      den Knoten eine 0 ausgegeben wird.     *
*  Eingabe-Parameter: KPTR: Pointer auf die zu bearbeitende  *
*                            Stelle im Baum.                  *
*  Return-Wert       : keiner                                 *
*************************************************************/
```

Listing 2: *(Fortsetzung)*

Inserentenverzeichnis

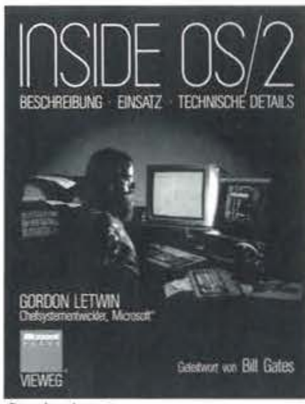
VIEWEG

COMPUTER BÜCHER UND SOFTWARE

FRÜHJAHR '89

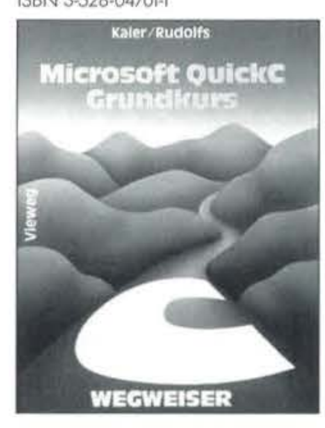


Augie Hansen
Ex-C-ellent
473 S. Geb. DM 98,–
ISBN 3-528-04592-2

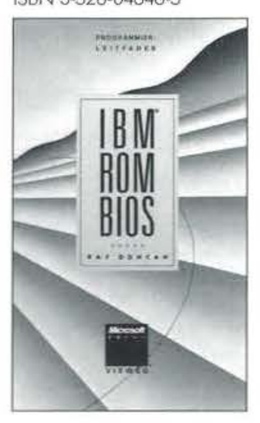


Gordon Letwin
Inside OS/2
376 S. Kart. DM 78,–
ISBN 3-528-04628-7

Ekkehard Kaier / Edwin Rudolfs
Microsoft Quick C Wegweiser Grundkurs
482 S. Kart. DM 54,–
ISBN 3-528-04701-1



Ray Duncan
Programmierleitfaden IBM ROM BIOS
133 S. Geb. DM 29,80
ISBN 3-528-04646-5



Ray Duncan
Programmierleitfaden MS-DOS Funktionen
148 S. Geb. DM 29,80
ISBN 3-528-04650-3



Ray Duncan (Hrsg.)
MS-DOS von A . . Z
306 S. Kart. DM 78,–
ISBN 3-528-04621-X





B. Rosemann, M. Kerres, D. J. Schlopsnies, H. Fink
MS-Excel
Mit diesem Buch wird Ihnen in leichtverständlichen Arbeitsschritten der Einstieg in das neuartige Planungssystem erleichtert. Im Anhang finden Sie Übungsaufgaben zu den einzelnen Kapiteln. Alle Übungsbeispiele mit den Lösungen sind auf der mitgelieferten Beispieldiskette enthalten.
1988, 183 Seiten, inkl. Diskette
Bestell-Nr. 90515
ISBN 3-89090-515-3
DM 69,– (sFr 63,50/öS 538,–)



R. Haselier/K. Fahnenstich
Programmieren mit Quick C
Dieses Buch zeigt Ihnen, wie Sie mit Quick C schnell und komfortabel eigene Programme erstellen können.
• Für den C-Einsteiger ebenso wie für den Fortgeschrittenen.
1988, 412 Seiten,
inkl. zwei Disketten
Bestell-Nr. 90609
ISBN 3-89090-609-5
DM 69,– (sFr 63,50/öS 538,–)



R. Haselier/K. Fahnenstich
Quick C Toolbox
Neben einer Beschreibung des Standards finden Sie alles, was Sie schon immer in Ihrer C-Bibliothek vermißt haben. Für Quick C, Version 1.01 und Microsoft C, Version 5.1.
Lieferbar 1. Quartal 1989,
ca. 200 Seiten,
inkl. drei 5¼"-Disketten
Bestell-Nr. 90674
ISBN 3-89090-674-5
ca. DM 98,–* (sFr 90,20*/öS 834,–*)



Dr. N. Meder/G. König/P. Scheuber **MS-OS/2**
Dieses erste Buch der Edition Microsoft gibt Ihnen – als erfahrenem Anwender oder PC-Software-Entwickler – einen Einstieg und einen Überblick über das neue Betriebssystem MS-OS/2!
Die Installation und die ersten Schritte werden ausführlich beschrieben. Den Schwerpunkt bildet der neue Befehlsvorrat von MS-OS/2.
1987, 304 Seiten
Bestell-Nr. 90512
ISBN 3-89090-512-9
DM 79,– (sFr 72,20/öS 616,–)



Dr. F. M. Sonner/M. Theis
MS-OS/2 für Software-Entwickler
Im ersten Teil des Buches werden Themenkomplexe behandelt wie Speicherverwaltung, Multitasking und Programmierschnittstelle. Im zweiten Teil finden Sie praktische Programmierbeispiele, und der dritte Teil enthält einen ausführlichen Referenzteil.
1988, 246 Seiten
Bestell-Nr. 90638
ISBN 3-89090-638-9
DM 79,– (sFr 72,20/öS 616,–)

* Unverbindliche Preisempfehlung

Markt & Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.



Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

```
void write_tree( KNOPER kptr )
{
 RG BYTE i, j;                                  /* Schleifenzähler */

 if ( kptr->typ == KNOTEN )   /* an einem Knoten angelangt? */
  {                                                     /* Ja */
   write_bit( 0 );                           /* 0-Bit ausgeben */
   write_tree( kptr->links );        /* Rekursion nach links */
   write_tree( kptr->rechts );      /* Rekursion nach rechts */
  }
 else                               /* an einem Blatt angelangt */
  {
   write_bit( 1 );                           /* 1-Bit ausgeben */

   /*-- ASCII-Code des Zeichens ausgeben ------------------*/

   for( j=kptr->ascii, i=8; i; --i, j >>= 1)
    write_bit( j & 1 );
  }
}

/*************************************************************
*  Funktion         : C O U N T _ O C C U R                  *
**---------------------------------------------------------**
*  Aufgabe          : Initialisiert den Baum und zählt die   *
*                     Häufigkeit innerhalb des angegebenen   *
*                     Speicherbereichs.                      *
*  Eingabe-Parameter: KPTR : Pointer auf den Baum-(Vektor).  *
*                     PTR  : Zeiger auf den ersten Help-Text *
*                            im Speicher                     *
*  Return-Wert      : keiner                                 *
*************************************************************/
void count_occur( KNOPER kptr, HPDP ptr )
{
 int       i;                                 /* Schleifenzäher */
 RG KNOPER nptr;                      /* Laufzeiger in den Baum */
 RG BPTR   lptr;               /* Laufzeiger in die Help-Texte */

 /*-- den Baum (noch in vektorieller Form) initialisieren --*/

 for (i=0, nptr=kptr; i<256; i++)   /* den Baum durchlaufen */
  {
   nptr->typ = BLATT;         /* jeder Eintrag ist ein Blatt */
   nptr->links = nptr->rechts = NIL;   /* keinen Nachfolger */
   nptr->anzahl= 0;    /* noch keine Häufigkeiten ermittelt */
   ( nptr++ )->ascii = i;          /* den ASCII-Code merken */
  }

 /*-- die einzelnen Help-Texte im Speicher durchlaufen -----*/

 while ( ptr != NIL )         /* alle Help-Texte bearbeitet? */
  {                                                   /* Nein */
   /*-- die Häufigkeit der Zeichen zählen -----------------*/

   for ( i = ptr->len, lptr = (BPTR) &ptr->i; i-- ; )
    ++( kptr + *(lptr++) )->anzahl;     /* Häufigkeit inkr. */
   ptr = ptr->next;          /* Zeiger auf nächsten Help-Text */

  }
}

/*************************************************************
*  Funktion         : G E T _ D I F F E R E N T              *
**---------------------------------------------------------**
*  Aufgabe          : Elemeniert die Einträge innerhalb des  *
*                     Baums, die Zeichen repräsnetieren, die *
*                     im Text gar nicht vorkommen.           *
*  Eingabe-Parameter: KPTR : Pointer auf den Baum.           *
*  Return-Wert      : die Anzahl der aufgetretenen Codes - 1 *
*************************************************************/
int get_different( KNOPER kptr )
{
 RG KNOPER aktptr,           /* Ptr zeigt auf aktuellen Node */
           mptr,    /* Ptr speichert der aktuellen Position */
           endptr;      /* Ptr zeigt auf das Ende der Liste */
 int       i;                               /* Schleifenzähler */

 endptr = aktptr = kptr + 255;  /*Pointer auf Ende der Liste*/
 while ( aktptr >= kptr )      /* Anfang der Liste erreicht? */
  if ( aktptr->anzahl )   /* tritt das Zeichen im Text auf? */
   --aktptr;            /* Ja, den nächsten Eintrag bearbeiten */
  else                /* das Zeichen tritt im Text nicht auf */
   {
    /*-- nächsten belegten Node suchen & Liste heranziehen -*/
    mptr = aktptr;                 /* aktuelle Position merken */
    for ( i=0;
          aktptr >= kptr  &&  aktptr->anzahl == 0;
          ++i, --aktptr)
     ;
    memcpy( aktptr+1, mptr+1, (endptr-mptr) * sizeof(NODE) );
    endptr -= i;      /* ENDPTR an hinter neues Ende setzen */
   }
 return( endptr - kptr ); /* Anzahl der Codes zurückliefern */
}

/*************************************************************
*  Funktion         : C O M P _ N O D E                      *
**---------------------------------------------------------**
*  Aufgabe          : Vergleicht zwei Elemente aus dem Baum. *
*  Eingabe-Parameter: EL1,                                   *
*                     EL2 : die zu vergleichenden Strukturen *
*  Return-Wert      : <0 wenn (EL1->anzahl) < (EL2->anzahl)  *
*                      0 wenn (EL1->anzahl) = (EL2->anzahl)  *
*                     >0 wenn (EL1->anzahl) > (EL2->anzahl)  *
*  Info             : Diese Funktion wird nur von der Funk-  *
*                     tion BUILD_TREE in Verbindung mit der  *
*                     Sortierung des Baums über QSORT auf-   *
*                     gerufen.                               *
*************************************************************/
```

Listing 2: *(Fortsetzung)*

```
int comp_node( const void * el1, const void * el2 )
{
 return ((KNOPER) el2)->anzahl - ((KNOPER) el1)->anzahl;
}

/*************************************************************
*  Funktion         : B U I L D _ T R E E                    *
**---------------------------------------------------------**
*  Aufgabe          : Erstellt den binären Baum, aus dem die *
*                     Codesequenzen für die einzelnen Zeichen*
*                     gewonnen werden können.                *
*  Eingabe-Parameter: KPTR : Pointer auf den Baum.           *
*  Return-Wert      : keiner                                 *
*************************************************************/

void build_tree( KNOPER kptr )
{
 int       diff_b,   /* Anzahl noch nicht eingegl. Elemente */
           i;                               /* Schleifenzähler */
 RG KNOPER rfollow,                      /* rechter Nachfolger */
           lfollow,                       /* linker Nachfolger */
           lptr;    /* Laufzeiger zum Durchsuchen des Baums */
 NODE      merke;           /* nimmt temporär einen NODE auf */

 /*-- Anzahl der Codes ermitteln und Baum nach aufstei-   --*/
 /*-- gender Häufigkeit der repräsentierten Zeichen sort. --*/

 diff_b = get_different( baum );
 qsort(kptr, diff_b+1, sizeof(NODE), comp_node );

 /*-- in der folgenden Schleife werden jeweils die letzten -*/
 /*-- beiden Einträge in der Liste zum linken und rechten  -*/
 /*-- Nachfolger eines Knotens zusammengefaßt.             -*/
 /*-- Danach wird die Summe der durch sie repräsentiereten -*/
 /*-- Zeichen ermittelt und der neue Knoten nach diesem    -*/
 /*-- Kriterium in absteigender Reihenfolge in den Baum    -*/
 /*-- einsortiert.                                         -*/
 for ( ; diff_b > 0 ; --diff_b )
  {
   /*- die beiden letzten Einträge in eine Struktur kopieren*/

   lfollow = (rfollow = (KNOPER) alloc(sizeof(NODE)*2)) + 1;
   *rfollow = *(kptr+diff_b);  /*letzter E. = rechter Nachf.*/
   *lfollow = *(kptr+diff_b-1);   /*vorl. E. = linker Nachf.*/

   merke.typ    = KNOTEN;          /* Eintrag ist ein Knoten */
   merke.links  = lfollow;      /* linken und rechten Nach- */
   merke.rechts = rfollow;      /* folger setzen             */
   merke.anzahl = rfollow->anzahl + lfollow->anzahl;

   /*-- neuen Eintrag in die Liste einsortieren -----------*/

   for (i=0, lptr=kptr;
        (i < diff_b-1) && (merke.anzahl < lptr->anzahl);
        ++i, ++lptr
       ) ;
   if ( i != diff_b )      /* wird Einträg hinten angehängt? */
    {                     /* Nein, Platz in der Liste schaffen */
     memmove(lptr+1, lptr, (diff_b - i) * sizeof(NODE) );
    }
   *lptr = merke;
  }
}

/*************************************************************
*  Funktion         : B U I L D _ T A B                      *
**---------------------------------------------------------**
*  Aufgabe          : Erstellt die Tabelle CODETAB, in der   *
*                     die Bitsequenzen für die einzelnen Bit-*
*                     sequenzen für die verschiedenne Zeichen*
*                     erfaßt werden.                         *
*  Eingabe-Parameter: KPTR : Pointer auf die aktuelle Posi-  *
*                            tion im Baum                    *
*                     CPTR : Pointer auf die Struktur, inner-*
*                            halt derer das Bitmuster für das*
*                            Zeichen erstellt wird.          *
*  Return-Wert      : keiner                                 *
*  Info             : Diese Funktion arbeitet sich rekursive *
*                     durch den Baum, es muß also genügend   *
*                     Speicher auf dem Stack zur Verfügung   *
*                     stehen.                                *
*************************************************************/
void build_tab( KNOPER kptr, UBES cptr )
{
 /*-- bereits an einem Blatt angelangt? ------------------*/

 if (kptr->typ == KNOTEN)
  {                /* Nein, durch Rekursion dem Blatt nähern */

   /*-- die Rekursion nach links weiterführen, wobei zu-  --*/
   /*-- nächst eine '0' an den Codestring angehangen wird --*/

   *(cptr.code+cptr.len++) = '0';
   build_tab( kptr->links, cptr );

   /*-- kehrt die Funktion aus der Rekursion nach links ----*/
   /*-- zurück, wird die letzte '0' im Codestring durch ----*/
   /*-- eine '1' ersetzt und die Rekursion dann nach     ----*/
   /*-- rechts fortgeführt.                              ----*/

   *(cptr.code+cptr.len-1) = '1';
   build_tab( kptr->rechts, cptr );
  }
 else                           /* Ja, an einem Blatt angelangt */
  {
   *(cptr.code+cptr.len) = '\0';      /* Codestring beenden */
   codetab[kptr->ascii] = cptr;         /* Codestring merken */
  }
}
```

Listing 2: *(Fortsetzung)*

```
/**************************************************************
*  Funktion          : H P D _ W R I T E                      *
**----------------------------------------------------------**
*  Aufgabe           : Schreibt eine bestimmte Anzahl von     *
*                      Bytes in die Help-Datei.               *
*  Eingabe-Parameter: BPTR = Zeiger auf den Puffer, aus dem  *
*                             die Daten übertragen werden     *
*                      ANZ  = Anzahl der zu schreibenden      *
*                             Bytes                           *
*                      FP   = Der zu der Datei gehörende File-*
*                             Pointer.                        *
*  Return-Wert       : keiner                                 *
*  Info              : Bei einem Ausgabe-Fehler kehrt die     *
*                      Funktion nicht zum Aufrufer zurück,    *
*                      sondern springt zur Fehlerbehandlungs- *
*                      routine.                               *
**************************************************************/

void hpd_write( void * bptr, unsigned anz, FILE * fp )
{
 fwrite( bptr, anz, 1, fp );                  /* Daten schreiben */
 if ferror( fp )                        /* Fehler beim Schreiben? */
  longjmp(err_jmp, E_WRITE);                               /* Ja */
}

/**************************************************************
*  Funktion          : C M P _ W R I T E                      *
**----------------------------------------------------------**
*  Aufgabe           : Komprimiert die einzelnen Help-Texte   *
*                      und legt den Vektor an, der später an  *
*                      den Anfang der kompilierten Help-Datei *
*                      geschrieben wird und die einzelnen     *
*                      Help-Texte innerhalb der Datei be-     *
*                      schreibt. Danach schreibt es die ver-  *
*                      schiedenen Datenblöcke in die neue     *
*                      Help-Datei.                            *
*  Eingabe-Parameter: DATPTR = Ptr auf den Dateinamen der    *
*                               zu erstellenden Datei         *
*  Return-Wert       : keiner                                 *
**************************************************************/

void cmp_write( char *datptr )
{
 RG BPTR treeptr,                   /* Zeiger auf Puffer mit Baum */
         bptr,     /* Dummy zum Zwischenspeichern eines Ptr */
         tptr,                 /* Zeiger auf temporären Puffer */
         ltptr;           /* Laufzeiger in temporären Puffer */
 int     tree_len,                       /* Länge des Codebaums */
         i, j, k;                       /* temporäre Variablen */
 HCDP    hcdp,          /* Zeiger auf Vektor mit Beschreibern */
         lhcdp;             /* Laufzeiger in den obigen Vektor */
 HPDP    lhpdp;       /* Laufzeiger in Help-Texte im Speicher */
 char  * code;   /* Zeiger auf das Bitmuster eines Zeichens */
 FILE  * fp;              /* File-Pointer für Ausgabedatei */
 struct header kopf;                  /* Kopf einer Help-Datei */


 /*-- Code-Baum in Puffer schreiben und länge ermitteln ----*/

 treeptr = (BPTR) alloc( 2048 ); /* vorsorglich 2 KB allok. */
 WB_INIT( treeptr ); /* Pufferadr. an WRITE BIT übermitteln */
 write_tree( baum );                /* Baum in Puffer schreiben */
 WB_ENDE();                             /* Ausgabe abschließen */
 tree_len = WB_GET_LEN();      /* Länge des Baums ermitteln */
 treeptr = realloc( treeptr, tree_len ); /* Puffer anpassen */

 /*-- Puffer für Vektor allokieren -------------------------*/
 lhcdp = hcdp = alloc( k = i = (anzhelp + 1) * sizeof(HCD) );

 /*-- die einzelnen Help-Text im Speicher durchlaufen, sie -*/
 /*-- komprimieren und dabei gleichzeitig Kenndaten in den -*/
 /*-- Vektor eintragen                                      -*/

 printf("\tKomprimierung Help-Text ..........    0");

 lhpdp = start_hpd;
 lhcdp->fpos = sizeof( struct header ) + tree_len + i;
 while ( lhpdp != NIL )   /* weiteren Help-Text bearbeiten? */
  {                                                   /* Ja */
   printf("\b\b\b\b\b%5d", lhcdp->hnr = lhpdp->nummer );

   /*-- temporären Puffer einrichten und Help-Text in ------*/
   /*-- diesen Puffer kopieren                         ------*/
   ltptr = tptr = alloc ( j = i = lhcdp->len = lhpdp->len );
   memcpy( tptr, bptr = (BPTR) &lhpdp->i, i );
   WB_INIT( bptr );

   while ( j-- )    /* die einzelnen Byte im Puffer durchl. */
    {               /* und Byte im original Puffer kodieren */
     code = (codetab + *ltptr++) -> code;  /*Bitmuster holen*/
     while ( *code )                 /* Bit für Bit ausgeben */
      write_bit( *code++ & 1 );
    }
   WB_ENDE();                              /* Puffer abschließen */
   free( tptr );                /* temporären Puffer freigeben */
   lhpdp = lhpdp->next;    /* Zeiger auf nächsten Help-Text */

   /*-- Dateiposition des nächsten Help-Texts setzen------- */
   (lhcdp+1)->fpos = lhcdp->fpos + WB_GET_LEN();

   ++lhcdp;             /* nächsten Vektor-Eintrag bearbeiten */
  }
 printf("\n");


 /*-- Help-Datei im Binär-Modus öffnen --------------------*/

 if ( !(fp = fopen( datptr, "wb" )) )
  longjmp(err_jmp, E_DEST);   /* kann nicht geöffnet werden */
```

Listing 2: *(Fortsetzung)*

```
 /*-- Dateikopf erstellen und ausgeben --------------------*/
 strcpy( kopf.str, START_STR );
 kopf.anzahl  = anzhelp;
 kopf.code    = ID_CODE;
 hpd_write( &kopf, sizeof kopf, fp );
 hpd_write( hcdp, k, fp );                /* Vektor ausgeben */
 hpd_write( treeptr, tree_len, fp );        /* Baum ausgeben */

 /*-- die einzelnen Help-Text im Speicher durchlaufen und --*/
 /*-- in die Datei schreiben                              --*/
 printf("\tAusgabe Help-Text ................    0");

 lhpdp = start_hpd;           /* Zeiger auf ersten Help-Text */
 while ( lhpdp != NIL )   /* weiteren Help-Text bearbeiten? */
  {     /* jeweils einen komprimierten Help-Text ausgeben */
   printf("\b\b\b\b\b%5d", hcdp->hnr );
   hpd_write( &lhpdp->i , (hcdp+1)->fpos - hcdp->fpos, fp );
   ++hcdp;                /* Ptr auf nächsten Vektor-Eintrag */
   lhpdp = lhpdp->next;    /* Zeiger auf nächsten Help-Text */
  }

 fclose( fp );                            /* Datei schließen */
 if ( ferror( fp ) )                               /* Fehler? */
  longjmp( err_jmp, E_CLOSE );                        /* Ja! */
}

/**************************************************************
*  Funktion          : H L P C _ D R I V E R                  *
**----------------------------------------------------------**
*  Aufgabe           : Koordiniert die einzelnen Schritte     *
*                      zur Kompilierung und Komprimierung der *
*                      Help-Datei sowie zur anschließenden    *
*                      Ausgabe.                               *
*  Eingabe-Parameter: ARGC,                                  *
*                      ARGV : wie beim MAIN()                 *
*  Return-Wert       : keiner                                 *
**************************************************************/
void hlpc_driver( int argc, char *argv[] )
{
 UBES codel;        /* beschreibt Bitmuster eines Zeichens */
 printf("\n\tBearbeitung Zeile ................    0");
 compile( argv[1] );                          /* Kompilieren */
 printf("\n\tErstellung Code-Tabelle........... ");
 count_occur( baum, start_hpd );        /* Häufigkeiten erm. */
 build_tree( baum );                          /* Baum aufbauen */
 codel.len = 0;          /* noch kein Codelement umgewandelt */
 build_tab( baum, codel );           /* Codetabelle aufbauen */
 printf(" O.K.\n");
 cmp_write( out_name( argv[1] ) );
 printf("\n");
}

/*************************************************************/
/**                      HAUPTPROGRAMM                     **/
/*************************************************************/

int main( int argc, char *argv[] )
{
 static char *err_mes[] =                  /* Fehlermeldungen */
  {
   "Die Source-Datei existiert nicht",
   "Fehler bei Lesen der Source-Datei",
   "Zu wenig Speicher",
   "Die Ziel-Datei kann nicht geöffnet werden",
   "Fehler bei Zugriff auf Ziel-Datei",
   "Fehler beim Schließen der Ziel-Datei",
   "Zeile zu lang",
   "Anfang einer Help-Seite erwartet",
   "Ungütlige Help-Nummer",
   "Help-Text ist leer",
   "Help-Text zu lang",
   "Unbekannter Befehlscode",
   "Verweise-Ende-Zeichen erwartet",
   "Punkt erwartet"
  };

 unsigned fehler;        /* nimmt Fehlercode von SETJMP auf */
 printf("\n             HELPCOMP - (c) 1989 by MICHAEL TIS"
        "CHER   \n");
 if ( argc!= 2 )              /* ungültige Anzahl Argumente? */
  {
   printf("\nHELPCOMP bereitet Dateien auf die Verarbeitung "
          "durch die\nSAA-Help-Engine vor.\n\nGeben Sie  daz"
          "u beim Aufruf des Programms  den Namen der\nzu be"
          "arbeitenden  Datei an. HELPCOMP erstellt aus dies"
          "er\nDatei eine SAA-Help-Datei mit der Erweiterung"
          " \".HLP\".\n\nWeitere Informationen  zu diesem Pr"
          "ogramm  finden Sie in\nder Juli-/August-Ausgabe d"
          "es Microsoft System Journal.\n");
   return( 1 );
  }
 else               /* es wurde genau ein Argument übergeben */
  {
   if ( (fehler = setjmp(err_jmp)) == 0 )
    {                            /* Installation von SETJMP */
     hlpc_driver( argc, argv );
     return( 0 );   /* Datei wurde ordnungsgemäß kompiliert */
    }
   else        /* LONGJMP wurde bei einem Fehler aufgerufen */
    {
     if ( fehler >= PRINT_NUM )          /* Zeile ausgeben? */
      printf("\nFehler in Zeile: %ld : %s.\n",          /* Ja */
             zeile, err_mes[ fehler ]);
     else                                             /* Nein */
      printf("\nFehler: %s.\n", err_mes[ fehler ]);
     return( 1 );               /* Fehlercode zurückliefern */
    }
  }
}
```

Listing 2: *(Ende)*

```
/*
[ ]------------------------------------------------------------[ ]
 |                                                              |
 |    Include-Datei     : HLP.H                                 |
 |    zur Einbindung    : der Funktionen zur Bildschirm-        |
 |                        verwaltung                            |
 |    erstellt am       : 19.05.1989                            |
 |    letztes Update am: 20.05.1989                             |
 |                                                              |
 |    (Copyright)       : 1989 by MICHAEL TISCHER               |
 |                                                              |
[ ]------------------------------------------------------------[ ]
*/

/*-- Typdeklarationen -----------------------------------------*/

#ifndef BOOL                          /* wie BOOLEAN in Pascal */
 typedef BYTE BOOL;
#endif

/*-- vordefinierte Konstanten für die Farbcodes der ver- ---*/
/*-- schiedenen Objekte innerhalb des Help-Fensters      ---*/

/*------------------------------ Farben im monochrom-Modus */
#define MON_N  NORMAL              /* normale Zeichen         */
#define MON_U  UNDERLINE           /* unterstrichene Zeichen */
#define MON_V  HNORMAL             /* Verweise                */
#define MON_A  INVERS              /* aktueller Verweis       */
#define MON_FR HNORMAL             /* Rahmenfarbe             */
#define MON_BU HNORMAL             /* Buttons                 */
#define MON_SL NORMAL              /* Slider                  */
#define MON_PF INVERS              /* Pfeil an Slidern        */

/*---------------------------------- Farben im Color-Modus */
#define COL_N  COL( WEISS, CYAN )      /* normale Zeichen   */
#define COL_U  COL( GELB, CYAN )       /* unterstr. Zeichen */
#define COL_V  COL( SCHWARZ, CYAN )    /* Verweise          */
#define COL_A  COL( CYAN, SCHWARZ )    /* aktueller Verweis */
#define COL_FR COL( BLAU, CYAN )       /* Rahmenfarbe       */
#define COL_BU COL( ROT, CYAN )        /* Buttons           */
#define COL_SL COL( CYAN, SCHWARZ )    /* Slider            */
#define COL_PF COL( CYAN, SCHWARZ)     /* Pfeil an Slidern  */

/*-- Position und Größe des Help-Fensters -----------------*/

#define HF_XO 5                                /* Startspalte */
#define HF_YO 5                                 /* Startzeile */
#define HF_X  64      /* Fensterbreite (inklusive Rahmen ) */
#define HF_Y  15       /* Fensterlänge (inklusive Rahmen ) */

/*-- Makros zum Setzen der Farben -------------------------*/

#define HlpSCHText( c ) ( col_n = c )    /* normaler H-Text */
#define HlpSCVer( c )   ( col_v = c )           /* Verweis */
#define HlpSCVerA( c )  ( col_a = c ) /* aktueller Verweis */
#define HlpSCUnd( c )   ( col_u = c )  /* unterstr. H-Text */
#define HlpSCFrame( c ) ( col_fr = c )     /* Rahmen um HF */
#define HlpSCBut( c )   ( col_bu = c )          /* Buttons */
#define HlpSCSlid( c )  ( col_sl = c )           /* Slider */
#define HlpSCPfeil( c ) ( col_pf = c ) /* Pfeile an Slidern */

/*-- externe Variablen -------------------------------------*/

extern BYTE col_n,           /* die versch. Farb-Variablen  */
            col_v,
            col_a,
            col_u,
            col_fr,
            col_bu,
            col_sl,
            col_pf;

/*-- Funktions-Deklarationen ------------------------------*/

BOOL HlpInit( char * hnptr  );
void HlpShow( int hnr );
void HlpSetHf( BYTE x0, BYTE y0, BYTE x, BYTE y );
```

Listing 3: `HLP.H`

```
/***************************************************************/
/*                          H L P . C                          */
/*-------------------------------------------------------------*/
/*    Aufgabe          : stellt eine Help-Engine in Verbin-    */
/*                       dung mit den verschiedenen Moduln     */
/*                       der SAA-Reihe zur Verfügung.          */
/*-------------------------------------------------------------*/
/*    Autor            : MICHAEL TISCHER                        */
/*    entwickelt am    : 14.05.1988                             */
/*    letztes Update   : 20.05.1988                             */
/*-------------------------------------------------------------*/
/*    Erstellung       : CL /A[S|M|C|L|H] HLP.C /C              */
/*                       dann mit einem anderen Modul linken   */
/***************************************************************/

/*== Include-Dateien einbinden ==============================*/

#include <stdlib.h>
#include <stdio.h>
```

Listing 4: `HLP.C`

```
#include <string.h>
#include <bios.h>
#include "vio.h"                      /* auch die SAA-Include- */
#include "kbm.h"                      /* Dateien einbinden     */
#include "men.h"
#include "hlpi.h"
#include "hlp.h"

/*== Makros ===============================================*/

#define StartByte()   ( akt_bit = 1 )   /* für HlpiGetBit() */

#define InitHSlider() ( hslider = 0 )
#define InitVSlider() ( vslider = 0 )
#define GetVSlider()  ( vslider )
#define GetHSlider()  ( hslider )

/*-- Zugriff auf den Mausbereich in Verbindung mit der -----*/
/*-- Funktion HlpiMouDelay()                           -----*/

#define SetMBer( m )  ( mber = m )
#define ResetMBer()   ( SetMBer( KEIN_BEREICH ) )
#define GetMBer()     ( mber )

/*== Typdeklarationen ======================================*/

typedef struct vdef VDEF;        /* Verweis-Descriptor      */
typedef VDEF *      VDP;         /* Zeiger auf einen solchen */
typedef union vel   VEL;                        /* aus VIO.H */
typedef VEL *       VELP;             /* Zeiger auf ein VEL */

/*== Strukturen und Unions ================================*/

struct vdef                        /* Beschreibt einen Verweis */
 {
  BYTE spalte,          /* Entfernung von der oberen linken */
       zeile,           /* Ecke des Help-Texts              */
       len;                /* Anzahl der Zeichen im Verweis */
  int  vnr;           /* Nummer des adressierten Help-Texts */
 };

/*== Konstanten ===========================================*/


/*-- Pausen für HlpiMouDelay() ----------------------------*/

#define SHORT_DELAY 1L              /* 1 Ticks = 1/18 Sekunde */
#define LONG_DELAY  9L              /* 9 Ticks = 1/2 Sekunde  */

#define VW_STR "<\x11┘=Verweis>"                /* Buttons im  */
#define HI_STR "<F1=Hilfe>"                    /* Help-Fenster */
#define BA_STR "<ESC=zurück>"

#define HELPRA_H EINRA /* Rahmen um Help-Fenster horizontal */
#define HELPRA_V EINRA                            /* vertikal */

#define LAST_HT -1               /* letzten Hilfstext anzeigen */
#define STACK_EMPTY 2        /* Help-Text der angezeigt wird, */
                            /* wenn der Help-Stack leer ist   */
#define KEIN_VERWEIS 255            /* kein aktueller Verweis */

#define S_PRED -1               /* Konstanten für HlpiSucheV() */
#define S_SUCC 1

#define ANZ_MERKE 20                 /* Rückblätterungs-Tiefe */

/*== globale Variablen ====================================*/

NODE   baum;                  /* Wurzel für den binären Baum */
BYTE   akt_bit = 1;           /* Bit-Zähler für HlpiGetBit() */
FILE * fp;   /* File-Pointer zum Zugriff auf die Help-Datei */
HCDP   hti_ptr; /* Vektor mit Informationen über Help-Texte */
int    htanz;    /* Anzahl der Help-Texte in der Help-Datei */
char * hlp_nam;                       /* Name der Help-Datei */
BYTE   hf_x0 = HF_X0,           /* Help-Fenster linke Spalte */
       hf_y0 = HF_Y0,            /* Help-Fenster obere Zeile */
       hf_x  = HF_X,      /* Help-Fenster Breite in Spalten */
       hf_y  = HF_Y;        /* Help-Fenster Länge in Zeilen */


/*-- globale Variablen im Zusammenhang mit dem jeweils -----*/
/*-- aktuellen Help-Text                               -----*/

VELP     velp;                  /* Zeiger auf den Textpuffer im */
VDP      vdp;               /* Vektor mit Verweisbeschreibern */
BEREICH mouvek[10];           /* Mausbereiche im Help-Fenster */
BYTE     hslider = 0,  /* Position des horizontalen Sliders */
         vslider = 0,    /* Position des vertikalen Sliders */
         mber;          /* Nummer des aktuellen Mausbereichs */

struct hpinfo hi;       /* Anzahl Verweise, Spalten & Zeilen */


/*-- Farbcodes für die Zeichen innerhalb eines Help-Texts, -*/
/*-- den Rahmen des Help-Fensters etc.                  ----*/

BYTE col_n,                                  /* normale Farbe */
     col_u,                         /* unterstrichene Zeichen */
     col_v,                                       /* Verweise */
     col_a,                              /* aktueller Verweis */
     col_fr,                        /* Rahmen um Help-Fenster */
     col_bu,                       /* Ok-, Hilfe-Button etc. */
     col_sl,          /* horizontaler und vertikaler Slider */
     col_pf;                          /* Pfeile an den Slidern */

/*- Help-Stack mit Nummern der letzten ANZ_MERKE Help-Texte */

int hstack[ ANZ_MERKE + 1 ];                    /* FIFO-Stack */
```

Listing 4: *(Fortsetzung)*

```
/**********************************************************
*  Funktion          : H l p i G e t B i t               *
**-------------------------------------------------------**
*  Aufgabe           : Holt ein Bit aus der Help-Datei.   *
*  Eingabe-Parameter: keine                               *
*  Return-Wert       : Der Wert des Bits                  *
**********************************************************/

BYTE HlpiGetBit()
{
 static int akt_byte;        /* das aktuell bearbeitete Byte */

 if (--akt_bit == 0)           /* in ein neues Byte wechseln */
  {                                                 /* Ja */
   akt_bit = 8;                           /* wieder im Bit 8 */
   akt_byte = fgetc( fp );  /* ein Byte aus der Datei holen */
  }
 akt_byte <<= 1;     /* um eine Position nach links shiften */

 return( *((BYTE *) &akt_byte+1) & 1 );  /*Bit zurückliefern*/
}


/**********************************************************
*  Funktion          : H l p i G e t 8 B i t             *
**-------------------------------------------------------**
*  Aufgabe           : Liest 8 aufeinanderfolgende Bits aus *
*                      der Help-Datei und wandelt sie in ein *
*                      Byte um.                            *
*  Eingabe-Parameter: keine                               *
*  Return-Wert       : Das eingelesene Byte               *
**********************************************************/

BYTE HlpiGet8Bit()
{
 int     retw;                          /* der Return-Wert */
 RG BYTE i;                             /* Schleifenzähler */

 for (retw = i = 0; i<8; ++i)          /* Byte-Code einlesen */
  {    /* jeweils ein Bit nach Bit 8 holen und dann shiften */
   *((BYTE *) &retw +1) = HlpiGetBit();
   retw >>= 1;
  }
 return retw;                          /* Byte zurückliefern */
}


/**********************************************************
*  Funktion          : H l p i B u i l d T r e e         *
**-------------------------------------------------------**
*  Aufgabe           : Erstellt den binären Baum, aus dem die *
*                      Codesequenzen für die einzelnen Zeichen*
*                      ermittelt werden können.            *
*  Eingabe-Parameter: BPTR : Pointer auf die aktuelle Posi- *
*                             tion im Baum                 *
*  Return-Wert       : keiner.                            *
**********************************************************/

void HlpiBuildTree( KNOPER bptr )
{
 RG KNOPER rfollow,                     /* rechter Nachfolger */
           lfollow;                     /* linker Nachfolger */
 BYTE      i;                            /* Schleifenzähler */
 int       ascii;                   /* ASCII-Code des Blattes */

 if ( HlpiGetBit() == 0)              /* ein weiterer Knoten? */
  {                                                 /* Ja */
   bptr->typ = KNOTEN;
   lfollow = (rfollow = (KNOPER) malloc(sizeof(NODE) * 2))+1;

   /*-- zunächst den Baum nach links rekursiv durchlaufen --*/
   HlpiBuildTree(bptr->links = lfollow);

   /*-- nach Abschluß der Rekursion nach links erfolgt je- -*/
   /*-- weils die Rekursion nach rechts                    -*/

   HlpiBuildTree(bptr->rechts = rfollow);
  }
 else                            /* an einem Blatt angelangt */
  {
   bptr->typ = BLATT;
   bptr->links = bptr->rechts = NIL;     /* kein Nachfolger */
   bptr->ascii = HlpiGet8Bit();  /*ASCII-Code holen & merken*/
  }
}


/**********************************************************
*  Funktion          : H l p i G e t B y t e             *
**-------------------------------------------------------**
*  Aufgabe           : Decodiert ein Byte aus der Help-Datei *
*  Eingabe-Parameter: keiner                              *
*  Return-Wert       : das dekodierte Byte                *
**********************************************************/

BYTE HlpiGetByte()
{
 RG KNOPER bptr;                        /* Zeiger in den Baum */

 bptr = &baum;           /* an der Wurzel des Baumes beginnen */

 /*-- Innerhalb des Baumes bis zu dem Blatt wandern, das ---*/
 /*-- die glesene Bit-Folge repräsentiert               ---*/

 while ( bptr->typ == KNOTEN )
  bptr = ( HlpiGetBit()==1 ) ? bptr->rechts : bptr->links;

 return( bptr ->ascii );        /* ASCII-Code zurückliefern */
}
```

Listing 4: *(Fortsetzung)*

```
/**********************************************************
*  Funktion          : H l p i B u i l d M o u V e k     *
**-------------------------------------------------------**
*  Aufgabe           : Baut den Maus-Vektor auf           *
*  Eingabe-Parameter: keiner                              *
*  Return-Wert       : keiner                             *
**********************************************************/

void HlpiBuildMouVek()
{
 #define SV  ( sizeof( VW_STR ) - 1 )
 #define SH  ( sizeof( HI_STR ) - 1 )
 #define SB  ( sizeof( BA_STR ) - 1 )

 RG BYTE i;   /* Schleifenzähler zum Laden der Maus-Pointer */

 /*-- Koordinaten Hilfe-Feld --------------------------------*/

 mouvek[ 0 ].x1 = hf_x0 + hf_x - 2 - SH + 1;
 mouvek[ 0 ].x2 = hf_x0 + hf_x - 2;
 mouvek[ 0 ].y1 = mouvek[ 0 ].y2 = hf_y0 + hf_y - 2;

 /*-- Koordinaten Verweis-Feld ------------------------------*/

 mouvek[ 1 ].x2 = mouvek[ 0 ].x1 - 2;
 mouvek[ 1 ].x1 = mouvek[ 0 ].x1 - 2 - SV + 1;
 mouvek[ 1 ].y1 = mouvek[ 1 ].y2 = hf_y0 + hf_y - 2;

 /*-- Koordinaten ESC-Feld ----------------------------------*/

 mouvek[ 2 ].x2 = mouvek[ 1 ].x1 - 2;
 mouvek[ 2 ].x1 = mouvek[ 1 ].x1 - 2 - SB + 1;
 mouvek[ 2 ].y1 = mouvek[ 2 ].y2 = hf_y0 + hf_y - 2;

 /*-- Koordinaten Help-Text ---------------------------------*/

 mouvek[ 3 ].x1 = hf_x0 + 1;
 mouvek[ 3 ].x2 = hf_x0 + hf_x - 2;
 mouvek[ 3 ].y1 = hf_y0 + 1;
 mouvek[ 3 ].y2 = hf_y0 + hf_y - 4;

 /*-- Koordinaten Pfeil links -------------------------------*/

 mouvek[ 4 ].x1 = mouvek[ 4 ].x2 = hf_x0 + 1;
 mouvek[ 4 ].y1 = mouvek[ 4 ].y2 = hf_y0 + hf_y - 3;

 /*-- Koordinanten horizontale Slider-Box -------------------*/

 mouvek[ 5 ].x1 = hf_x0 + 2;
 mouvek[ 5 ].x2 = hf_x0 + hf_x - 3;
 mouvek[ 5 ].y1 = mouvek[ 5 ].y2 = hf_y0 + hf_y - 3;

 /*-- Koordinaten Pfeil rechts ------------------------------*/

 mouvek[ 6 ].x1 = mouvek[ 6 ].x2 = hf_x0 + hf_x - 2;
 mouvek[ 6 ].y1 = mouvek[ 6 ].y2 = hf_y0 + hf_y - 3;

 /*-- Koordinaten Pfeil oben --------------------------------*/

 mouvek[ 7 ].x1 = mouvek[ 7 ].x2 = hf_x0 + hf_x - 1;
 mouvek[ 7 ].y1 = mouvek[ 7 ].y2 = hf_y0 + 1;

 /*-- Koordinaten vertikale Slider-Box ----------------------*/

 mouvek[ 8 ].x1 = mouvek[ 8 ].x2 = hf_x0 + hf_x - 1;
 mouvek[ 8 ].y1 = hf_y0 + 2;
 mouvek[ 8 ].y2 = hf_y0 + hf_y - 5;

 /*-- Koordinaten Pfeil unten -------------------------------*/

 mouvek[ 9 ].x1 = mouvek[ 9 ].x2 = hf_x0 + hf_x - 1;
 mouvek[ 9 ].y1 = mouvek[ 9 ].y2 = hf_y0 + hf_y - 4;

 /*-- alle Bereiche werden mit einem einheitlichen Maus- ---*/
 /*-- Cuirsor versehen                                   ---*/

 for ( i = 0; i < 10 ; ++i )
  mouvek[ i ].ptr_mask = MouPtrMask( PTRSAMECHAR, PTRINVCOL );
}


/**********************************************************
*  Funktion          : H l p i M o u D e l a y           *
**-------------------------------------------------------**
*  Aufgabe           : Stoppt die Programmausführung für ei- *
*                      nen kurzen Moment, damit sich das Mo- *
*                      dul gegenüber der Maus nicht zu sensi- *
*                      tiv verhält.                        *
*  Eingabe-Parameter: MOUB = aktueller Mausbereich        *
*  Return-Wert       : keiner                             *
**********************************************************/

void HlpiMouDelay( BYTE moub )
{
 long ticks,                       /* zu erwartende Uhrzeit */
      akt;                              /* aktuelle Uhrzeit */

 /*-- aktuell Uhrzeit holen und Ziel-Uhrzeit berechnen -----*/
 _bios_timeofday( _TIME_GETCLOCK, &ticks );
 ticks += ( moub == GetMBer() ? SHORT_DELAY : LONG_DELAY );


 /*-- warten, bis die Zeit verstrichen ist -----------------*/
 do
  _bios_timeofday( _TIME_GETCLOCK, &akt );
 while ( akt < ticks );

 SetMBer( moub );                  /* neuen Mausbereich merken */
}
```

Listing 4: *(Fortsetzung)*

```
/****************************************************************
*  Funktion          : H l p i H S l i d e r                    *
**------------------------------------------------------------**
*  Aufgabe           : Markiert die aktuelle Spaltenposition    *
*                      im horizontalen Slider.                  *
*  Eingabe-Parameter: SPALTE = neue Spaltenposition am linken*
*                               Rand des Help-Fensters          *
*  Return-Wert       : keiner                                   *
****************************************************************/

void HlpiHSlider( BYTE spalte )
{
 VioPrint( VL( 2 + hslider ), VU(-2), col_sl, FALSE, "█" );
 if ( hi.spalten <= hf_x - 2 )          /* Text schmaler als HF? */
  hslider = 0;                          /* Ja, Slider in Spalte 0 */
 else                        /* Nein, Slider-Position berechnen */
  hslider = spalte * ( hf_x - 5 ) / ( hi.spalten - hf_x + 2 );
 VioPrint( VL( 2 + hslider ), VU(-2), col_sl, FALSE, " " );
}

/****************************************************************
*  Funktion          : H l p i V S l i d e r                    *
**------------------------------------------------------------**
*  Aufgabe           : Markiert die aktuelle Zeilenposition     *
*                      im vertikalen Slider.                    *
*  Eingabe-Parameter: ZEILE = neue Zeilenposition am oberen    *
*                              Rand des Help-Fensters           *
*  Return-Wert       : keiner                                   *
****************************************************************/

void HlpiVSlider( BYTE zeile )
{
 VioPrint( VR(0), VO( 2 + vslider ), col_sl, FALSE, "█" );
 if ( hi.zeilen <= hf_y - 4 )            /* Text kürzer als HF? */
  vslider = 0;                          /* Ja, Slider in Zeile 0 */
 else                        /* Nein, Slider-Position berechnen */
  vslider = zeile * ( hf_y - 7 ) / ( hi.zeilen - hf_y + 3 );
 VioPrint( VR(0), VO( 2 + vslider ), col_sl, FALSE, " " );
}

/****************************************************************
*  Funktion          : H l p i V i F                            *
**------------------------------------------------------------**
*  Aufgabe           : Stellt fest, ob sich ein Verweis ganz    *
*                      oder zumindest teilweise innerhalb des *
*                      Help-Fensters befindet.                  *
*  Eingabe-Parameter: VNR     = Nummer des Verweises            *
*                      SPALTE, = Zeichen innerhalb des Help-    *
*                      ZEILE     Texts, das in der oberen       *
*                                linken Ecke des Fensters       *
*                                dargstellt wird.               *
*  Return-Wert       : TRUE, wenn sich der Verweis innerhalb    *
*                      des Help-Fensters befindet, sonst FALSE*
****************************************************************/

BYTE HlpiViF( BYTE vnr, BYTE spalte, BYTE zeile )
{
 int      diffz,                          /* Differenz in Zeilen */
          diffs;                         /* Differenz in Spalten */
 RG VDP   lvdp;            /* Laufzeiger in den Verweis-Vektor */

 diffz = (lvdp = vdp + vnr )->zeile - zeile;
 if ( diffz >= 0  &&  diffz < hf_y-4 )
  {                          /* Verweis-Zeile innerhalb des HF */
   diffs = lvdp->spalte - spalte;
   if (!(diffs>=(int) hf_x-2 || (diffs+ (int)lvdp->len) < 1))
    return TRUE;                      /* Verweis innerhalb des HF */
  }
 return FALSE;          /* Verweis außerhalb des Help-Fensters */
}

/****************************************************************
*  Funktion          : H l p i S u c h e V                      *
**------------------------------------------------------------**
*  Aufgabe           : Sucht den nächsten Verweis innerhalb     *
*                      des Help-Fensters.                       *
*  Eingabe-Parameter: VNR     = Nummer des aktuellen Verw.     *
*                      OFFSET  = Suchrichtung                   *
*                      SPALTE, = Zeichen innerhalb des Help-    *
*                      ZEILE     Texts, das in der oberen       *
*                                linken Ecke des Fensters       *
*                                dargstellt wird.               *
*  Return-Wert       : die Nummer des neuen Verweises oder      *
*                      KEIN_VERWEIS, wenn kein Verweis ge-      *
*                      funden wurde.                            *
*  Info              : Für SUCHE können die Konstanten S_PRED *
*                      ( suche Vorgänger ) und S_SUCC ( suche *
*                      Nachfolger ) übergeben werden            *
****************************************************************/

BYTE HlpiSucheV(int vnr, int offset, BYTE spalte, BYTE zeile)

{
 /*-- den Verweis-Vektor ausgehend vom akt. Verw. durchl. --*/

 for (vnr+=offset; vnr>=0  &&  vnr < hi.verweise; vnr+=offset)
  if ( HlpiViF( vnr, spalte, zeile ) )       /* Verw. im HF? */
   return vnr;                      /* Ja, Nummer zurückliefern */

 return KEIN_VERWEIS;                 /* keinen Verweis entdeckt */
}

/****************************************************************
*  Funktion          : H l p i C o l o r V                      *
**------------------------------------------------------------**
*  Aufgabe           : Färbt einen Verweis innerhalb des        *
*                      Help-Puffers und innerhalb des Help-     *
*                      Fensters ein, soweit er dort angezeigt *
*                      wird.                                    *
```

Listing 4: *(Fortsetzung)*

```
*  Eingabe-Parameter: VNR     = Nummer des Verweises            *
*                      FARBE   = neue Farbe                     *
*                      SPALTE, = Zeichen innerhalb des Help-    *
*                      ZEILE     Texts, das in der oberen       *
*                                linken Ecke des Fensters       *
*                                dargstellt wird.               *
*  Return-Wert       : keiner                                   *
****************************************************************/

void HlpiColorV(int vnr, BYTE farbe, BYTE spalte, BYTE zeile)
{
 RG VELP lv;            /* Laufzeiger in den Help-Text-Puffer */
 VDEF    verweis;              /* Informationen über Verweis */
 BYTE    i;                                /* Schleifenzähler */
 RG int  diffz,                          /* Differenz in Zeilen */
         starsp,                /* Startspalte auf Bildschirm */
         endsp;                   /* Endspalte auf Bildschirm */

 verweis = *( vdp + vnr );               /* Verweis-Daten holen */
 lv = velp + verweis.zeile * hi.spalten + verweis.spalte;

 /*-- Verweis innerhalb des Help-Text-Puffers einfärben ----*/
 for ( i = verweis.len; i-- ; )
  lv++->h.attribut = farbe;

 diffz = verweis.zeile - zeile;
 if ( diffz >= 0  &&  diffz < hf_y-4 ) /* Zeile im Fenster? */
  {                                                    /* Ja */
   starsp = verweis.spalte - spalte;           /* Startspalte */
   endsp  = starsp + verweis.len - 1;            /* Endspalte */
   if ( !( starsp >= (int) hf_x-2  ||  endsp < 0 ) )
    {  /*zumindest ein Teil des Verweises ist im HF sichtbar*/
     if ( starsp < 0 )          /* Startspalte außerhalb HF? */
      starsp = 0;          /* Ja, mit der Spalte 0 beginnen */
     if ( endsp >= hf_x-2 )      /* Endspalte außerhalb HF? */
      endsp = hf_x-3;          /* Ja, am Ende des HF aufhören */
     VioColor( VL( 1 + starsp ), VO( 1 + diffz ),
               VL( 1 + endsp ), VO( 1 + diffz ), farbe );
    }
  }
}

/****************************************************************
*  Funktion          : H l p i S c r o l l D o w n              *
**------------------------------------------------------------**
*  Aufgabe           : Scrollt den Inhalt des Help-Fensters     *
*                      um eine Zeile nach unten.                *
*  Eingabe-Parameter: SPALTE, = Zeichen innerhalb des Help-    *
*                      ZEILE     Texts, das in der oberen       *
*                                linken Ecke des Fensters       *
*                                dargstellt wird.               *
*  Return-Wert       : keiner                                   *
****************************************************************/
void HlpiScrollDown( BYTE spalte, BYTE zeile )
{
 RG VELP lvelp;         /* Laufzeiger in den Help-Text-Puffer */
 RG BYTE endsp;                          /* Endspalte für Text */

 MouHideMouse();                        /* Maus-Cursor ausblenden */

 VioScrollDown( VL(1), VO(1), VR(-1), VU(-4), 1, NOCLEAR );
 lvelp = velp + ( zeile - 1 ) * hi.spalten + spalte;
 if ( spalte + hf_x - 1 <= hi.spalten )          /* Freiraum? */
  VioPut( VL(1), VO(1), VR(-1), VO(1), lvelp );        /* Nein */
 else
  {
   endsp = VL( hi.spalten - spalte );
   VioPut( VL(1), VO(1), endsp, VO(1), lvelp );
   VioClear( endsp+1, VO(1), VR(-1), VO(1), col_n );
  }

 HlpiVSlider( zeile-1 );     /* horizontalen Slider anzeigen */
 MouShowMouse();             /* Maus-Cursor wieder einblenden */
}

/****************************************************************
*  Funktion          : H l p i S c r o l l U p                  *
**------------------------------------------------------------**
*  Aufgabe           : Scrollt den Inhalt des Help-Fensters     *
*                      um eine Zeile nach oben.                 *
*  Eingabe-Parameter: SPALTE, = Zeichen innerhalb des Help-    *
*                      ZEILE     Texts, das in der oberen       *
*                                linken Ecke des Fensters       *
*                                dargstellt wird.               *
*  Return-Wert       : keiner                                   *
*  Info              : Es wird davon ausgegangen, daß sich in *
*                      der neuen letzten Bildschirmzeile Text *
*                      befindet.                                *
****************************************************************/
void HlpiScrollUp( BYTE spalte, BYTE zeile )
{
 RG VELP lvelp;         /* Laufzeiger in den Help-Text-Puffer */
 RG BYTE endsp;                          /* Endspalte für Text */

 MouHideMouse();                        /* Maus-Cursor ausblenden */

 VioScrollUp( VL(1), VO(2), VR(-1), VU(-3), 1, NOCLEAR );
 lvelp = velp + ( zeile + hf_y-4 ) * hi.spalten + spalte;
 if ( spalte + hf_x - 1 <= hi.spalten )          /* Freiraum? */
  VioPut( VL(1), VU(-3), VR(-1), VU(-3), lvelp );     /* Nein */
 else
  {
   endsp = VL( hi.spalten - spalte );
   VioPut( VL(1), VU(-3), endsp, VU(-3), lvelp );
   VioClear( endsp+1, VU(-3), VR(-1), VU(-3), col_n );
  }
 HlpiVSlider( zeile+1 );     /* horizontalen Slider anzeigen */
 MouShowMouse();             /* Maus-Cursor wieder einblenden */
}
```

Listing 4: *(Fortsetzung)*

```
/*************************************************************
*  Funktion          : H l p i S c r o l l L e f t           *
**-----------------------------------------------------------**
*  Aufgabe           : Scrollt den Inhalt des Help-Fensters  *
*                      um eine Spalte nach links             *
*  Eingabe-Parameter: SPALTE, = Zeichen innerhalb des Help-  *
*                     ZEILE     Texts, das in der oberen     *
*                               linken Ecke des Fensters     *
*                               dargstellt wird.             *
*  Return-Wert       : keiner                                *
*  Info              : Es wird davon ausgegangen, daß sich in *
*                      der neuen letzten Bildschirmspalte    *
*                      Text befindet.                        *
*************************************************************/

void HlpiScrollLeft( BYTE spalte, BYTE zeile )
{
 RG VELP lvelp;        /* Laufzeiger in den Help-Text-Puffer */
 RG BYTE i,                              /* Schleifenzähler */
         endsp;                                /* Endspalte */

 MouHideMouse();                    /* Maus-Cursor ausblenden */

 endsp = VR( -1 );                   /* Endspalte berechnen */
 VioScrollLeft( VL(2), VO(1), endsp, VU(-3), 1, NOCLEAR );
 lvelp = velp + zeile * hi.spalten + spalte + hf_x - 2;

 /*-- die einzelnen Zeilen des Fensters durchlaufen-------- */
 for ( i = VO( 1 );
       i < VU(-2)  &&  zeile < hi.zeilen;
       ++i, ++zeile)
  {             /* Zeichen am rechten Fenster-Rand ausgeben */
   VioPut( endsp, i, endsp, i, lvelp );
   lvelp += hi.spalten;           /* Zeiger auf nächste Zeile */
  }

 HlpiHSlider( spalte+1 );   /* horizontalen Slider anzeigen */
 MouShowMouse();           /* Maus-Cursor wieder einblenden */
}

/*************************************************************
*  Funktion          : H l p i S c r o l l R i g h t         *
**-----------------------------------------------------------**
*  Aufgabe           : Scrollt den Inhalt des Help-Fensters  *
*                      um eine Spalte nach rechts            *
*  Eingabe-Parameter: SPALTE, = Zeichen innerhalb des Help-  *
*                     ZEILE     Texts, das in der oberen     *
*                               linken Ecke des Fensters     *
*                               dargstellt wird.             *
*  Return-Wert       : keiner                                *
*************************************************************/

void HlpiScrollRight( BYTE spalte, BYTE zeile )
{
 RG VELP lvelp;        /* Laufzeiger in den Help-Text-Puffer */
 RG BYTE i,                              /* Schleifenzähler */
         starsp;                             /* Startspalte */

 MouHideMouse();                    /* Maus-Cursor ausblenden */

 starsp = VL( 1 );                   /* Endspalte berechnen */
 VioScrollRight( starsp, VO(1), VR(-2) , VU(-3), 1, NOCLEAR );
 lvelp = velp + zeile * hi.spalten + spalte - 1;

 /*-- die einzelnen Zeilen des Fensters durchlaufen-------- */
 for ( i = VO( 1 );
       i < VU( -2 )  &&  zeile < hi.zeilen;
       ++i, ++zeile)
  {                /* Zeichen am linken Rand des HF ausgeben */
   VioPut( starsp, i, starsp, i, lvelp );
   lvelp += hi.spalten;           /* Zeiger auf nächste Zeile */
  }

 HlpiHSlider( spalte-1 );   /* horizontalen Slider anzeigen */
 MouShowMouse();           /* Maus-Cursor wieder einblenden */
}

/*************************************************************
*  Funktion          : H l p i S h o w                       *
**-----------------------------------------------------------**
*  Aufgabe           : Baut den Help-Text innerhalb des Help- *
*                      Fensters ab einer bestimmten Position *
*                      komplett neu auf.                     *
*  Eingabe-Parameter: SPALTE, = Position, ab der der Help-   *
*                     ZEILE     Text dargestellt werden soll *
*  Return-Wert       : keiner                                *
*************************************************************/
void HlpiShow( BYTE spalte, BYTE zeile )
{
 RG VELP lvelp;        /* Laufzeiger in den Help-Text-Puffer */
 RG BYTE iz,                      /* Schleifenzähler Zeile */
         aktz,                 /* aktuelle Bildschirmzeile */
         starsp, /* Startspalte in Bezug auf den Bildschirm */
         endsp,    /* Endspalte in Bezug auf den Bildschirm */
         starbl,                 /* Startspalte für Blanks */
         endbl;                    /* Endspalte für Blanks */

 MouHideMouse();                    /* Maus-Cursor ausblenden */
 HlpiHSlider( spalte );     /* horizontalen Slider anzeigen */
 HlpiVSlider( zeile );        /* vertikalen Slider anzeigen */

 /*-- Zeiger in Help-Text-Puffer berechnen -----------------*/
 lvelp = velp + zeile * hi.spalten + spalte;
 starsp = VL( 1 );                  /* Startspalte berechnen */
 if ( spalte + hf_x - 1 <= hi.spalten )        /* Freiraum? */
  {                                                /* Nein */
   endsp  = VR( - 1 );
   starbl = 0;                                /* keine Blanks */
  }
 else                     /* Ja, Freiraum am Ende jeder Zeile */
  {
   endsp  = VL( hi.spalten - spalte );
   starbl = endsp + 1;   /* Blanks schließen an den Text an */
   endbl  = VR( -1 );               /* Endspalte der Blanks */
  }

 /*-- die einzelnen Zeilen innerhalb des Fensters durchl. --*/
 starsp = VL( 1 );                  /* Startspalte berechnen */
 for ( aktz = VO(1), iz = hf_y-4;
       iz--  &&  zeile < hi.zeilen;
       ++aktz, ++zeile )
  {
   VioPut( starsp, aktz, endsp, aktz, lvelp );
   if ( starbl )                        /* Blanks ausgeben? */
    VioClear( starbl, aktz, endbl, aktz, col_n );      /* Ja */
   lvelp += hi.spalten;
  }
 if ( aktz <= VU( -3 ) )      /* Leerzeilen am Fensterende? */
  VioClear( starsp, aktz, VR(-1) , VU(-3), col_n );   /* Ja */

 MouShowMouse();           /* Maus-Cursor wieder einblenden */
}

/*************************************************************
*  Funktion          : H l p S e t H f                       *
**-----------------------------------------------------------**
*  Aufgabe           : Legt die Position des Help-Fensters   *
*                      auf dem Bildschirm sowie seine Größe  *
*                      fest.                                 *
*  Eingabe-Parameter: X0 = linke Spalte                      *
*                     Y0 = obere Zeile                       *
*                     X  = Breite in Spalten                 *
*                     Y  = Länge in Zeilen                   *
*  Return-Wert       : keiner                                *
*************************************************************/

void HlpSetHf( BYTE x0, BYTE y0, BYTE x, BYTE y )
{
 RG VELP lv;                      /* Laufzeiger in Leerzeile */
 RG BYTE i;                               /* Schleifenzähler */

 hf_x0 = x0;               /* die übergebene Parameter in  */
 hf_y0 = y0;               /* die korrespondierenden Vari- */
 hf_x = x;                 /* ablen laden                  */
 hf_y = y;

 HlpiBuildMouVek();                 /* Maus-Vektor aufbauen */
}

/*************************************************************
*  Funktion          : H l p i B u i l d W i n               *
**-----------------------------------------------------------**
*  Aufgabe           : Baut das Help-Fenster auf             *
*  Eingabe-Parameter: keine                                  *
*  Return-Wert       : keiner                                *
*************************************************************/

void HlpiBuildWin()
{
 RG BYTE i;                               /* Schleifenzähler */

 MouHideMouse();                   /* Maus-Cursor ausblenden */
 MenWinOpen( hf_x0, hf_y0, hf_x0 + hf_x - 1,
             hf_y0 + hf_y - 1, HELPRA_H, HELPRA_V, col_fr );
 VioPrint( VL(2),  VU(-2), col_sl, FALSE,
           VioStrep( '█', hf_x-4 ) ) ;
 VioPrint( VL( 1), VU(-2), col_pf, FALSE, "\x1b" );
 VioPrint( VR(-1), VU(-2), col_pf, FALSE, "\x1A" );
 VioPrint( VR( 0), VU(-3), col_pf, FALSE, "\x19" );
 VioPrint( VR( 0), VO( 1), col_pf, FALSE, "\x18" );
 for ( i = VO( 2 ); i < VU( -3 ); ++i )
  VioPrint( VR(0),  i, col_sl, FALSE, "█" );

 VioPrint(mouvek[0].x1, mouvek[0].y1, col_bu, FALSE, HI_STR);
 VioPrint(mouvek[1].x1, mouvek[1].y1, col_bu, FALSE, VW_STR);
 VioPrint(mouvek[2].x1, mouvek[2].y1, col_bu, FALSE, BA_STR);

 MouShowMouse();             /* Maus-Cursor wieder anzeigen */
}

/*************************************************************
*  Funktion          : H l p i E r r o r                     *
**-----------------------------------------------------------**
*  Aufgabe           : Baut einen Help-Text aus einem einzei- *
*                      ligen Fehlertext auf.                 *
*  Eingabe-Parameter: SPTR = Pointer auf den Fehler-String   *
*  Return-Wert       : keiner                                *
*  Info              : - Die Adresse des Help-Text-Puffers   *
*                        wird in der globalen Variablen  VELP *
*                        abgelegt.                           *
*************************************************************/
void HlpiError( char * sptr )
{
 RG VELP lv;           /* Laufzeiger in den Help-Text-Puffer */
 BYTE    starsp,                   /* Startspalte des Texts */
         breite;                    /* Breite des Fensters */
 RG int  len,                   /* Länge des Fehlerstrings */
         i;                               /* Schleifenzähler */

 hi.zeilen = ( hf_y - 4  >> 1) + 1;  /*Text belegt halbes HF*/
 /*-- je nach der Breite des HF wird der Text zentriert ----*/
 if ( ( breite = len = strlen( sptr ) ) < hf_x - 2 )
  breite = ( starsp = hf_x - 2 - len >> 1 ) + len;
 else                   /* Nein, Fenster schmaler als Text */
  starsp = 0;                     /* Text beginnt in Spalte 0 */

 hi.spalten = breite;              /* Breite des Textes merken */
 hi.verweise = 0;               /* kein Verweis im Hilfstext */
```

Listing 4: *(Fortsetzung)*

Listing 4: *(Fortsetzung)*

```
 /*-- Puffer allokieren und zunächst mit Leerz. füllen -----*/
 lv = velp = malloc((i = breite * hi.zeilen ) * sizeof(VEL));
 while ( i-- )           /* den gesamten Puffer durchlaufen */
  {
   lv->h.attribut = col_n;                /* normale Farbe */
   lv++->h.zeichen = ' ';                  /* Leerzeichen */
  }

 /*-- den Fehlerstring durchlaufen und in Puffer schr. -----*/
 for (lv=velp+(hi.zeilen-1)*hi.spalten+starsp ; len-- ; )
  {
   lv->h.attribut = col_u; /* Zeichen besonders hervorheben */
   lv++->h.zeichen = *sptr++;  /* Zeichen aus Str. kopieren */
  }
}

/************************************************************
*  Funktion          : H l p i R e a d                      *
**--------------------------------------------------------**
*  Aufgabe           : Liest einen Help-Text ein und baut    *
*                      ihn in einem Puffer auf.             *
*  Eingabe-Parameter: HNR   = Nummer des Help-Texts         *
*  Return-Wert       : keiner                               *
*  Info              : - Die Adresse des Help-Text-Puffers   *
*                        wird in der globalen Variablen  VELP *
*                        abgelegt.                          *
*                      - Die Adresse des Vektors mit den Ver- *
*                        weisbeschreibern wird in der globalen*
*                        Variablen HVDP abgelegt.           *
************************************************************/

void HlpiRead( int hnr )
{
 #define LO(p) *((BPTR) &p )          /* LO-Byte eines INTs */
 #define HI(p) *((BPTR) &p + 1)       /* HI-Byte eines INTs */
 #define PUTCH( c ) ( lv->h.zeichen = c, \
                      lv++->h.attribut = aktcol, \
                      ++asp )

 char errstr[ MAX_Z_LEN + 1 ];                /* Fehlerstring */
 RG VELP lv;                 /* Laufzeiger in den Textpuffer */
 RG BPTR bptr;                     /* Zeiger in BYTE-Puffer */
 HCDP    lptr;                      /* Laufzeiger in Vektor */
 int     i;                             /* Schleifenzähler */
 BYTE    b,                /* das jeweils eingelesene Byte */
         aktcol,       /* die jeweils aktuelle Zeichenfarbe */
         asp,                          /* aktuelle Spalte */
         az,                            /* aktuelle Zeile */
         len;                     /* Länge eines Verweises */
 VDP     lvdp;                /* Laufzeiger in diesen Vektor */

 /*-- Help-Text im Help-Text-Vektor suchen -----------------*/
 for ( i=htanz, lptr=hti_ptr;
       i  &&  lptr->hnr != hnr;
       --i, ++lptr ) ;

 if ( i == 0 )                                 /* gefunden? */
  {                                                 /* Nein */
   sprintf( errstr, "ACHTUNG! Help-Text %d nicht verfügbar.",
            hnr );
   HlpiError( errstr );                /* Fehlertext aufbauen */
   return;                             /* zurück zum Aufrufer */
  }


 /*-- Dateizeiger auf Anfang des Help-Texts innerhalb der --*/
 /*-- Datei setzen und dann Struktur HI einlesen          --*/

 fseek( fp, lptr->fpos, SEEK_SET );
 StartByte();
 for ( i = sizeof hi, bptr = (BPTR) &hi; i-- ; )
  *bptr++ = HlpiGetByte();

 if ( ferror( fp ) )           /* Fehler beim Dateizugriff? */
  {                                                   /* Ja */
   sprintf( errstr,
            "ACHTUNG! Lesefehler bei Zugriff auf \"%s\".",
            hlp_nam );
   HlpiError( errstr );                /* Fehlertext aufbauen */
   return;                             /* zurück zum Aufrufer */
  }

 /*-- Vektor für Verweisinformationen allokieren -----------*/
 if ( hi.verweise )                     /* gibt es Verweise? */
  lvdp = vdp = malloc( hi.verweise * sizeof( VDEF ) );  /*Ja*/

 /*-- Vektor für Help-Text-Puffer allokieren ---------------*/
 aktcol = col_n;      /* Zeichen zunächst in normaler Farbe */
 lv = velp = malloc(hi.zeilen * hi.spalten * sizeof(VEL));


 /*-- die einzelnen Zeichen des Help-Texts durchlaufen -----*/

 asp = az = 0;                   /* aktuelle Zeile und Spalte */
 for ( i = lptr->len - sizeof hi ; i-- ; )
  switch( b = HlpiGetByte() )                /* Zeichen holen */
   {                                         /* und auswerten */
    case TOK_UNTER :                /* Zeichen unterstreichen */
     aktcol = col_u;                     /* neue Farbe setzen */
     break;

    case TOK_NORMAL :          /* normale Zeichendarstellung */
     aktcol = col_n;                     /* neue Farbe setzen */
     break;

    case TOK_CODE :                             /* Zeichen folgt */
     PUTCH( HlpiGetByte() );
     --i;            /* nächstes Zeichen wurde schon gelesen */
     break;
```

Listing 4: *(Fortsetzung)*

```
    case TOK_ZENDE :                            /* Zeilenende */
     aktcol = col_n;         /* auf normale Farbe umschalten */
     while ( asp != hi.spalten )     /* Rest der Zeile mit */
      PUTCH( ' ' );                    /* Spaces belegen    */
      asp = 0;                /* wieder in Spalte 0 beginnen */
     ++az;                          /* Zeile inkrementieren */
     break;

    case TOK_VERWEIS :              /* einen Verweis entdeckt */
     aktcol = col_v;                         /* Verweis-Farbe */
     len = 0;             /* Länge des Verweises bisher 0 */
     lvdp->spalte = asp;               /* Koordinate merken */
     lvdp->zeile = az;
     do                                       /* Leseschleife */
      {
       --i;                       /* ein Byte weniger lesen */
       switch( b = HlpiGetByte() )          /* Byte auswerten */
        {
         case TOK_CODE :                       /* Zeichen folgt */
          ++len;             /* Verweis-Länge inkrementieren */
          PUTCH( HlpiGetByte() );
          --i;       /* nächstes Zeichen wurde schon gelesen */
          break;

         case TOK_VERWEIS :                 /*  nichts passiert */
          break;

         default :                    /* jedes andere Zeichen */
          ++len;             /* Verweis-Länge inkrementieren */
          PUTCH( b );         /* Zeichen in Textpuffer bringen */
          break;
        }
      }
     while ( b != TOK_VERWEIS );
     lvdp->len = len;            /* Länge des Verweises merken */
     LO(lvdp->vnr) = HlpiGetByte();  /* LO-Byte Verwweis-Nr */
     HI(lvdp->vnr) = HlpiGetByte();   /* Hi-Byte Verweis-Nr */
     ++lvdp;              /* LVDP auf nächsten Eintrag setzen */
     i -= 2;                      /* zwei Bytes mehr gelesen */
     aktcol = col_n;         /* auf normale Farbe umschalten */
     break;

    default :                         /* jedes andere Zeichen */
      PUTCH( b );             /* Zeichen in Textpuffer bringen */
      break;
   }
}

/************************************************************
*  Funktion          : H l p i T r a c e H S l i d e r       *
**--------------------------------------------------------**
*  Aufgabe           : Bewegt den horizontalen Slider hinter *
*                      dem Maus-Cursor her.                 *
*  Eingabe-Parameter: SPTR = Ptr auf Variable mit akt. Spalte*
*                      ZPTR = Ptr auf Variable mit akt. Zeile *
*  Return-Wert       : keiner                               *
************************************************************/
void HlpiTraceHSlider( BPTR sptr, BPTR zptr )
{
 RG int  event;                             /* Maus-Ereignis */
 RG BYTE spalte,   /* Spalte, in der sich die Maus befindet */
         newsp;                                /* neue Spalte */

 spalte = VL( 2 + hslider );       /* aktuelle Slider-Spalte */
 while ( TRUE )                           /* Abfrageschleife */
  {                                /* Maus-Ereignis abfragen */
   event = KbmEventWait( EV_MOU_MOVE | EV_LEFT_REL );
   if ( event & EV_MOU_MOVE )              /* Maus bewegt? */
    {                                                  /* Ja */
     if ( ( newsp = MouGetCol() ) < VL(2) )
      newsp = VL( 2 );  /* Maus-Cursor links von Slider-Box */
     else
      if ( newsp > VR( - 2 ) )
       newsp = VR( - 2 );     /* Maus rechts von Slider-Box */

     if ( newsp != spalte )                  /* Veränderung? */
      {                                                /* Ja */
       spalte = newsp;                  /* neue Spalte merken */
       MouHideMouse();             /* Maus-Cursor ausblenden */
       VioPrint( VL(2+hslider), VU(-2), col_sl, FALSE, "█");
       hslider = newsp - VL( 2 );
       VioPrint( VL(2+hslider), VU(-2), col_sl, FALSE, " " );
       MouShowMouse();        /* Maus-Cursor wieder anzeigen */
      }
    }
   else                 /* linker Mausknopf wurde losgelassen */
    break;                            /* WHILE-Schleife beenden */
  }
 if ( hi.spalten < hf_x - 2 )       /* Text schmaler als HF? */
  HlpiHSlider( 0 );         /* Ja, Slider-Bewegung ignorieren */
 else         /* neue Spaltenposition durch Slider-Bewegung */
  {
   if ( spalte == VR( -2 ) )   /* am r. Rand d. Slider-Box? */
    spalte = hi.spalten - ( hf_x - 2 );                /* Ja */
   else          /* Spalte aus Slider-Position berechnen */
    spalte = hslider * (hi.spalten - hf_x + 2) / (hf_x - 5);
   HlpiShow( *sptr = spalte, *zptr );  /* Text neu aufbauen */
  }
}

/************************************************************
*  Funktion          : H l p i T r a c e V S l i d e r       *
**--------------------------------------------------------**
*  Aufgabe           : Bewegt den vertikalen Slider hinter   *
*                      dem Maus-Cursor her.                 *
*  Eingabe-Parameter: SPTR = Ptr auf Variable mit akt. Spalte*
*                      ZPTR = Ptr auf Variable mit akt. Zeile *
*  Return-Wert       : keiner                               *
************************************************************/
```

Listing 4: *(Fortsetzung)*

```
void HlpiTraceVSlider( BPTR sptr, BPTR zptr )
{
 RG int  event;                             /* Maus-Ereignis */
 RG BYTE zeile,     /* Zeile, in der sich die Maus befindet */
         newz;                             /* neue Maus-Zeile */

 zeile = VO( 2 + vslider );         /* aktuelle Slider-Zeile */
 while ( TRUE )                           /* Abfrageschleife */
  {                               /* Maus-Ereignis abfragen */
   event = KbmEventWait( EV_MOU_MOVE | EV_LEFT_REL );
   if ( event & EV_MOU_MOVE )                /* Maus bewegt? */
    {                                               /* Ja */
     if ( ( newz = MouGetRow() ) < VO(2) )
      newz = VO( 2 );         /* Maus-Cursor über Slider-Box */
     else
      if ( newz > VU( -4 ) )
       newz = VU( -4 );            /* Maus unter Slider-Box */

     if ( newz != zeile )                    /* Veränderung? */
      {                                               /* Ja */
       zeile = newz;                   /* neue Zeile merken */
       MouHideMouse();             /* Maus-Cursor ausblenden */
       VioPrint( VR(0), VO(2+vslider), col_sl,FALSE, "█");
       vslider = newz - VO( 2 );
       VioPrint( VR(0), VO(2+vslider), col_sl, FALSE, " " );
       MouShowMouse();        /* Maus-Cursor wieder anzeigen */
      }
    }
   else              /* linker Mausknopf wurde losgelassen */
    break;                         /* WHILE-Schleife beenden */
  }
 if ( hi.zeilen < hf_y - 4 )            /* Text kürzer als HF? */
  HlpiVSlider( 0 );        /* Ja, Slider-Bewegung ignorieren */
 else        /* neue Spaltenposition durch Slider-Bewegung */
  {
   if ( zeile == VU( -4 ) )     /* am u. Rand d. Slider-Box? */
    zeile = hi.zeilen - ( hf_y - 4 );                  /* Ja */
   else         /* Nein, Zeile aus Slider-Position berechnen */
    zeile = vslider * (hi.zeilen - hf_y + 3) / (hf_y - 7);
   HlpiShow( *sptr, *zptr = zeile );  /* Help-Text neu auf. */
  }
}

/***************************************************************
*  Funktion          : H l p i P r o c e s s                  *
**------------------------------------------------------------**
*  Aufgabe           : Lädt eine Help-Text ein, zeigt ihn auf *
*                      dem Bildschirm an und verwaltet die    *
*                      Eingaben von Maus und Tastatur.        *
*  Eingabe-Parameter: HNR   = Nummer des Help-Texts           *
*  Return-Wert       : Nummer des nächsten Help-Texts oder 0, *
*                      wenn kein Help-Text mehr angezeigt     *
*                      werden soll.                           *
***************************************************************/

int HlpiProcess( int hnr )
{
 #define SUCHV( v, o ) HlpiSucheV( v, o, spalte, zeile )
 #define COLV( v, c )  HlpiColorV( v, c, spalte, zeile )

 RG VDP  lvdp;                  /* Zeiger in Verweis-Vektor */
    BYTE i,                           /* für diverse Zwecke */
         spalte,  /* Zeichens innerhalb des Help-Texts, das */
         zeile,   /* in der oberen linken Fensterecke steht */
         nsp,                                /* neue Spalte */
         nzl,                                 /* neue Zeile */
         anzl,              /* Anzahl Help-Zeilen im Fenster */
         anzsp,            /* Anzahl Help-Spalten im Fenster */
         aktver,                       /* aktueller Verweis */
         mouz,                                /* Maus-Zeile */
         mous;                               /* Maus-Spalte */
 RG int  event,     /* Ereignis von Tastatur und/oder Maus */
         neuver,               /* neuer aktueller Verweis */
         nexthp;                      /* nächster Help-Text */
 TASTE   key;                   /* Zeichen von der Tastatur */
 BOOL    ende;

 HlpiRead( hnr );          /* Help-Text einladen & dekodieren */
 InitHSlider();           /* die beiden Slider initialisieren */
 InitVSlider();
 ResetMBer();                       /* Mausbereich zurücksetzen */
 anzl  = hf_y - 4;         /* Anzahl sichtbarer Help-Zeilen */
 anzsp = hf_x - 2;        /* Anzahl sichtbarer Help-Spalten */
 ende  = FALSE;
 aktver = KEIN_VERWEIS;     /* noch keinen aktuellen Verweis */
 HlpiShow( nexthp = spalte = zeile = 0, 0 );


 /*-- Interaktionsschleife ---------------------------------*/
 while ( !ende )
  {
   if ( aktver == KEIN_VERWEIS )        /* z.Zt kein Verweis? */
    {                          /* Nein, einen Verweis suchen */
     aktver = SUCHV( -1, S_SUCC );
     if ( aktver != KEIN_VERWEIS )       /* Verweis gefunden? */
      COLV( aktver, col_a );                           /* Ja */
    }
   else                /* aktueller Verweis noch im Fenter? */
    if ( !HlpiViF( aktver, spalte, zeile ) )
     {                                               /* Nein */
      COLV( aktver, col_v );  /* wieder Verweis-Farbe verl. */
      neuver = SUCHV( aktver, S_PRED );   /*Vorgänger suchen*/
      if ( neuver == KEIN_VERWEIS )              /* gefunden?*/
       aktver = SUCHV( aktver, S_SUCC ); /* Nein, Nachf. s. */
      else           /* es wurde ein Vorgänger gefunden */
       aktver = neuver;     /* Nr. des neuen Verweises laden */
      if ( aktver != KEIN_VERWEIS ) /* jetzt einen Verweis? */
       COLV( aktver, col_a );                /* Ja, markieren */
     }
```

Listing 4: *(Fortsetzung)*

```
   event = KbmEventWait( EV_LEFT_PRESS | EV_KEY_AVAIL );
   if ( event & EV_KEY_AVAIL )               /* Taste betätigt? */
    {                                 /* Ja, Taste auswerten */
     switch( key = KbdGetKey() )
      {
       case CHOME :          /*----------------- Cursor-Home */
        if ( !(spalte == 0  &&  zeile == 0 ) )
         HlpiShow( spalte = zeile = 0, 0 );
        break;
       case CEND :           /*------------------ Cursor End */
        if ( hi.zeilen < anzl )
         nzl = 0;
        else
         nzl = hi.zeilen - anzl;
        if ( !( nzl == zeile  &&  spalte == 0) )
         HlpiShow( spalte = 0, zeile = nzl );
        break;

       case CDOWN :          /*----------------- Cursor Down */
        if ( zeile + anzl < hi.zeilen )
         HlpiScrollUp( spalte, zeile++ );
        break;

       case CUP :            /*------------------- Cursor Up */
        if ( zeile )                 /* Zeile noch nicht Null? */
         HlpiScrollDown( spalte, zeile-- );             /* Nein */
        break;

       case CLEFT :                            /* Cursor Left */
        if ( spalte )               /* Spalte noch nicht Null? */
         HlpiScrollRight( spalte--, zeile );            /* Nein */
        break;

       case CRIGHT :         /*---------------- Cursor Right */
        if ( spalte + anzsp < hi.spalten )
         HlpiScrollLeft( spalte++, zeile );
        break;

       case CPGDN :          /*------------------- Page-Down */
        if ( zeile + anzl + anzl < hi.zeilen )
         HlpiShow( spalte, zeile += anzl );    /* PgDn mögl. */
        else     /* Page-Down nicht möglich, Textende anz. */
         HlpiShow( spalte, zeile = hi.zeilen - anzl );
        break;

       case CPGUP :          /*--------------------- Page-Up */
        if ( zeile >= anzl )                /* Pg-Up möglich? */
         HlpiShow( spalte, zeile -= anzl );                /* Ja */
        else     /* Page-Up nicht möglich, Textanfang anz. */
         HlpiShow( spalte, zeile = 0 );                    /* Ja */
        break;

       case TAB :            /*------------ nächsten Verweis */
         if ( aktver != KEIN_VERWEIS )    /* Verweis im HF? */
          {                                               /* Ja */
           neuver = aktver;   /* neuer Verw. ist auch alter */
           if ((neuver=SUCHV(aktver,S_SUCC)) == KEIN_VERWEIS)
            neuver = SUCHV( -1, S_SUCC );
           if ( neuver != aktver && neuver != KEIN_VERWEIS )
            {                         /* neuen Verweis gefunden */
             COLV( aktver, col_v );   /* alten Verw. ausbl. */
             COLV( aktver = neuver, col_a );
            }
          }
         break;

       case BACKTAB :          /*-- vorhergehender Verweis */
         if ( aktver != KEIN_VERWEIS )    /* Verweis im HF? */
          {                                               /* Ja */
           neuver = aktver;   /* neuer Verw. ist auch alter */
           if ((neuver=SUCHV(aktver,S_PRED)) == KEIN_VERWEIS)
            neuver = SUCHV( hi.verweise, S_PRED );
           if ( neuver != aktver  && neuver != KEIN_VERWEIS )
            {                         /* neuen Verweis gefunden */
             COLV( aktver, col_v );   /* alten Verw. ausbl. */
             COLV( aktver = neuver, col_a );
            }
          }
         break;

       case F1:          /*--------------- Hilfe über Hilfe */
        ende = TRUE;                      /* Funktion beenden */
        nexthp = HILFE_TEXT;       /* Nummer des Help-Texts */
        break;

       case ALT_F1:        /*-- letzten Hilfstext anzeigen */
        ende = TRUE;                      /* Funktion beenden */
        nexthp = LAST_HT; /* Konstante: "letzten Help-Text" */
        break;

       case CR:            /*---------- Hilfstext anwählen */
        ende = TRUE;
        nexthp=(aktver==KEIN_VERWEIS) ? 0 : (vdp+aktver)->vnr;
        break;

       case ESC :          /*------------- Hilfe verlassen */
        ende = TRUE;
        nexthp = 0;       /* keinen Help-Text mehr anzeigen */
        break;
      }
    }
   else                  /* linken Mausknopf niedergedrückt */
    {
     switch ( MouGetBereich() )    /* Mausbereich auswerten */
      {
       case 0 :            /*---------------- Hilfe-Button */
        ende = TRUE;                      /* Funktion beenden */
        nexthp = HILFE_TEXT;   /* Hilfe über Hilfe anzeigen */
        break;
```

Listing 4: *(Fortsetzung)*

```
    case 1 :             /*--------------- Verweis-Button */
     ende = TRUE;                     /* Funktion beenden */
     nexthp=(aktver==KEIN_VERWEIS) ? 0 : (vdp+aktver)->vnr;
     break;

    case 2 :             /*--------------- Escape-Button */
     ende = TRUE;                     /* Funktion beenden */
     nexthp = 0;       /* keinen Hilfstext mehr anzeigen */
     break;

    case 3 :             /*---- innerhalb des Help-Texts */
      mouz = MouGetRow() - VO(1) + zeile ; /* Text-Zeile */
      mous = MouGetCol() - VL(1) + spalte;  /*Text-Spalte*/
      /*-- befindet sich die Maus auf einem Verweis?---- */
      for ( i=hi.verweise, lvdp = vdp; i; ++lvdp, --i )
       if ( mouz == lvdp->zeile  &&
            mous >= lvdp->spalte &&
            mous <  lvdp->spalte + lvdp-> len )
        break;                        /* Verweis gefunden */
      if ( i )            /* wurde ein Verweis gefunden? */
       {                                          /* Ja */
        nexthp = lvdp->vnr;  /*Nummer des nächsten H-Text*/
        ende = TRUE;                  /* Funktion beenden */
       }
      break;

    case 4 :             /*----------------- Pfeil links */
     if ( spalte )            /* Spalte noch nicht Null? */
      HlpiScrollRight( spalte--, zeile );        /* Nein */
     break;

    case 5 :             /*--------- horizontaler Slider */
      i = MouGetCol() - VL( 2 );          /* rel. Position */
      if ( i < GetHSlider() )       /* links vom Slider? */
       {             /* Ja, identisch mit Pfeil-links */
        if ( spalte )         /* Spalte noch nicht Null? */
         HlpiScrollRight( spalte--, zeile );     /* Nein */
       }
      else                         /* rechts vom Slider? */
       if ( i > GetHSlider() )
        {           /* Ja, identisch mit Pfeil-rechts */
         if ( spalte + anzsp < hi.spalten )
          HlpiScrollLeft( spalte++, zeile );
        }
       else                           /* auf dem Slider */
        HlpiTraceHSlider( &spalte, &zeile );
      break;

    case 6 :             /*---------------- Pfeil rechts */
     if ( spalte + anzsp < hi.spalten )
      HlpiScrollLeft( spalte++, zeile );
     break;

    case 7:              /*------------------ Pfeil oben */
     if ( zeile )              /* Zeile noch nicht Null? */
      HlpiScrollDown( spalte, zeile-- );         /* Nein */
     break;

    case 8 :             /*----------- vertikaler Slider */
      i = MouGetRow() - VO( 2 );          /* rel. Position */
      if ( i < GetVSlider() )        /* über dem Slider? */
       {              /* Ja, identisch mit Pfeil-oben */
        if ( zeile )          /* Zeile noch nicht Null? */
        HlpiScrollDown( spalte, zeile-- );       /* Nein */
       }
      else                         /* unter dem  Slider? */
       if ( i > GetVSlider() )
        {           /* Ja, identisch mit Pfeil-unten */
         if ( zeile + anzl < hi.zeilen )
          HlpiScrollUp( spalte, zeile++ );
        }
       else                           /* auf dem Slider */
        HlpiTraceVSlider( &spalte, &zeile );
      break;

    case 9:              /*----------------- Pfeil unten */
     if ( zeile + anzl < hi.zeilen ) /* wieter scrollen? */
      HlpiScrollUp( spalte, zeile++ );             /* Ja */
     break;
   }
  HlpiMouDelay( MouGetBereich() );   /* Maus-Verzögerung */
  }
 }

 /*-- die in HlpiRead allokierten Puffer wieder freigeben --*/
 if ( hi.verweise )                  /* gibt es Verweise? */
  free( vdp );                                     /* Ja */
 free( velp );

 HlpiHSlider( 0 );            /* die zwei Slider ausblenden */
 HlpiVSlider( 0 );

 return nexthp;  /* Nummer des nächsten Help-Texts zurückl. */
}

/************************************************************
*  Funktion         : H l p S h o w                         *
**---------------------------------------------------------**
*  Aufgabe          : Steuert den Aufbau eines Help-Texts   *
*                     sowie die Abfrage von Maus & Tastatur. *
*  Eingabe-Parameter: HNR   = Nummer des Help-Texts         *
*  Return-Wert      : keiner                                *
************************************************************/

void HlpShow( RG int hnr )
{
 MouPushPara();                    /* Maus-Parameter sichern */
 HlpiBuildWin();                    /* Help-Fenster aufbauen */
 MouSetBereich( ELVEK( mouvek ), mouvek ); /*Mausber. setzen*/
```

Listing 4: *(Fortsetzung)*

```
 do
  {
   /*-- Platz im Help-Stack schaffen ----------------------*/
   memmove( &hstack[1], &hstack[0],
            sizeof hstack - sizeof hstack[0] );
   hstack[0] = hnr;                   /* Help-Nummer merken */

   hnr = HlpiProcess( hnr );          /* Help-Text anzeigen */
   if ( hnr == LAST_HT )      /* letzten Help-Text anzeigen */
    {                                                /* Ja */
     hnr = hstack[1];       /* Help-Nummer aus Stack holen */
     /*-- Help-Stack um eine Position nach vorne schieben --*/
     memmove( &hstack[0], &hstack[2],
              sizeof hstack - ( 2 * sizeof hstack[0] ) );
     hstack[ ANZ_MERKE-1 ] = STACK_EMPTY;
    }
  }
 while ( hnr );

 MouHideMouse();                     /* Maus-Cursor ausblenden */

 MenWinClose( TRUE );                /* Help-Fenster schließen */

 MouShowMouse();               /* Maus-Cursor wieder anzeigen */

 MouPopPara();                /* Maus-Parameter zurückholen */
}


/************************************************************
*  Funktion         : H l p I n i t                         *
**---------------------------------------------------------**
*  Aufgabe          : Leitet die Arbeit mit dem Help-Modul  *
*                     ein und legt den Namen der Help-Datei *
*                     fest.                                 *
*  Eingabe-Parameter: HNPTR : Pointer auf einen String, der *
*                             den Namen der Help-Datei ent- *
*                             hält.                         *
*  Return-Wert      : TRUE, wenn die Datei vorgefunden und  *
*                     einwandfrei initialisiert werden konn- *
*                     te, sonst FALSE                       *
************************************************************/

BOOL HlpInit( char * hnptr  )
{
 RG int   vlen,                    /* Länge des HCD-Vektors */
        * hstptr;               /* Zeiger in den Help-Stack */
 BYTE     i;                              /* Schleifenzähler */
 struct header kopf;                   /* Kopf der Help-Datei */


 if ( ( fp = fopen( hlp_nam = hnptr, "rb" ) ) == NULL )
  return FALSE;        /* Datei konnte nicht geöffnet werden */

 if ( fread( &kopf, sizeof kopf, 1, fp ) != 1 )
  return FALSE;          /* Kopf konnte nicht gelesen werden */

 if ( kopf.code != ID_CODE )                   /* Help-Datei? */
  return FALSE;               /* Nein, Code nicht gefunden */


 htanz = kopf.anzahl;       /* Anzahl der Help-Texte merken */
 hti_ptr = malloc( vlen = ( htanz + 1 ) * sizeof( HCD ) );
 if ( fread( hti_ptr, 1, vlen, fp ) != vlen )
  return FALSE;        /* Vektor konnte nicht gelesen werden */

 HlpiBuildTree( &baum );                 /* Codebaum aufbauen */


 /*-- Help-Stack initialisieren ---------------------------*/

 for ( i = ANZ_MERKE+1, hstptr = hstack; i-- ; )
  *hstptr++ = STACK_EMPTY;


 /*-- Farben für die verschiedenen Zeichen festlegen -------*/

 if ( VioIsColor() )                      /* im Color-Modus? */

  {                                                  /* Ja */
   HlpSCHText( COL_N );                 /* normaler H-Text */
   HlpSCVer( COL_V );                           /* Verweis */
   HlpSCVerA( COL_A );               /* aktueller Verweis */
   HlpSCUnd( COL_U );                 /* unterstr. H-Text */
   HlpSCFrame( COL_FR );                 /* Rahmen um HF */
   HlpSCBut( COL_BU );                          /* Buttons */
   HlpSCSlid( COL_SL );                          /* Slider */

   HlpSCPfeil( COL_PF );              /* Pfeile an Slidern */
  }

 else                                 /* im monochrom-Modus */

  {                                                  /* Ja */
   HlpSCHText( MON_N );                 /* normaler H-Text */
   HlpSCVer( MON_V );                           /* Verweis */
   HlpSCVerA( MON_A );               /* aktueller Verweis */
   HlpSCUnd( MON_U );                 /* unterstr. H-Text */
   HlpSCFrame( MON_FR );                 /* Rahmen um HF */
   HlpSCBut( MON_BU );                          /* Buttons */
   HlpSCSlid( MON_SL );                          /* Slider */
   HlpSCPfeil( MON_PF );              /* Pfeile an Slidern */
  }

 HlpiBuildMouVek();                    /* Maus-Vektor aufbauen */

}
```

Listing 4: *(Ende)*

```
/*************************************************************/
/*                    H L P D E M O . C                      */
/*-----------------------------------------------------------*/
/*    Aufgabe         : demonstriert die Arbeit mit der SAA- */
/*                      Help-Endigne                         */
/*-----------------------------------------------------------*/
/*    Autor           : MICHAEL TISCHER                      */
/*    entwickelt am   : 14.05.1988                           */
/*    letztes Update  : 20.05.1988                           */
/*-----------------------------------------------------------*/
/*    Erstellung      : CL /A[S|M|C|L|H] HLPDEMO.C HLP.C     */
/*                      VIO.C KBM.C MEN.C MENUTIL.C          */
/*************************************************************/

#include "vio.h" /* die SAA-Include-Dateien einbinden  */
#include "kbm.h"
#include "hlp.h"

/*== Hauptprogramm =*/

void main( int argc, char *argv[] )
{
 VioInit(); /* Video-Modul initialisieren */
 KbmInit(); /* Tastatur- und Maus-Modul initialisieren */
 VioClearScreen( VioIsColor() ? COL( WEISS, BLAU ) : INVERS );
 MouShowMouse(); /* Maus-Cursor anzeigen */
 if ( HlpInit( "hlpdemo.hlp" ) ) /* Help-Modul initialisieren */
  HlpShow( 2 ); /* Help-Text Nummer 2 anzeigen */
 else /* Help-Datei nicht gefunden */
  printf("ACHTUNG! Fehlende oder falsche Help-Datei.\n");
}
```

Listing 5: HLPDEMO.C

```
; HLPDEMO.TXT eine Help-Demo-Datei
; (c) 1989 by MICHAEL TISCHER
; --  1 : grundsätzlich Hilfe über Hilfe
:1
                      Hilfe über Hilfe
─────────────────────────────────────────────────────────────

    Hom  \x18 |Pup║   Sofern  der Help-Text  größer  als
                   das  Help-Fenster ist,  läßt  sich
    \x1b | 5 | \x1a ║   der sichtbare Ausschnitt des Help-
                   Texts  mit Hilfe der Cursor-Tasten
    End  \x19 |Pdn║   verschieben.

    T A B   und   SHIFT   T A B    springen zum je-
                                   weils   nächsten
                                   Verweis

    ALT   F1   bringt den zuletzt betrachteten Help
               Text wieder zurück

    \x11─┘  ║ holt den aktuell markierten  Verweis auf
            den Bildschirm

    E S C   zurück zum Programm

                \x11\v... über die Help-Engine\v2.\x10
; --  2 : erscheint, wenn der Help-Stack leer ist
:2
                   über die SAA-Help-Engine
─────────────────────────────────────────────────────────────

 Das Help-Modul  bietet Ihnen  im Rahmen  der  SAA-Serie  die
 Möglichkeit, beliebige  Help-Texte zu  erstellen und  sie in
 Ihre  SAA-Applikationen   einzubinden.  Mit  vergleichsweise
 geringem Aufwand können Sie dem Anwender dadurch den Komfort
 professioneller Anwendungen  bieten. Der  Entwicklungsprozeß
 umfaßt dabei nur ganze vier Schritte:

   ■ Erstellen Sie  die verschiedenen  Help-Texte  innerhalb
     einer  Help-Datei  mit  einem  beliebigen  Editor  oder
     Textverarbeitungsprogramm und speichern Sie diese Datei
     als unformatiertes ASCII-File ab.

   ■ Kompilieren  Sie   diese  Help-Datei   mit  Hilfe   des
     Programms »HLPCOMP«  und bringen Sie sie dadurch in ein
     für die SAA-Help-Engine lesbares Format.

   ■ Fügen  Sie   einen  Aufruf   der   Funktion   »HlpInit(
     Dateiname)« in  den Startcode  Ihres Programms  an, und
     rufen Sie bei der Betätigung der entsprechenden (Help-)
     Taste innerhalb  Ihres Programms die Funktion »HlpShow(
     Nr )« auf, um den Help-Text mit der Nummer »Nr« auf den
     Bildschirm zu holen.

   ■ Kompilieren Sie  das Help-Modul  »HLP.C« und  verbinden
     Sie es mit den anderen SAA-Moduln und Ihrem Programm.

          \x11\vAufbau einer Help-Datei\v3.\x10  \x11\vDemo-Seite\v5.\x10
```

Listing 6: HLPDEMO.T

```
;
;
; -- 3 : Aufbau einer Help-Datei
;
:3
                  Aufbau einer Help-Datei
─────────────────────────────────────────────────────────────

 Eine  Help-Datei,   wie  sie   dem  Programm  »HLPCOMP«  zur
 Kompilierung  übergeben  wird,  muß  ein  bestimmtes  Format
 aufweisen, damit ihre Bestandteile von »HLPCOMP« erkannt und
 verarbeitet werden können.

 Die einzelnen Help-Texte müssen dazu in aufeinanderfolgenden
 Zeilen angesiedelt  und jeweils  durch eine Zeile, beginnend
 mit einem  Doppelpunkt, separiert werden. Diesem Doppelpunkt
 muß die Nummer des Help-Texts (in dezimaler Form) folgen. Es
 sind Kennummern  zwischen 1  und 32000  erlaubt, wobei jeder
 Help-Text eine  individuelle Nummer  tragen muß. Bis auf die
 Forderung, daß  der Help-Text  1 grundsätzlich Informationen
 über  die  Arbeit  mit  der  Hilfe-Funktion  enthalten  muß,
 unterliegt  die   Numerierung   der   Help-Texte   keinerlei
 Ristriktionen.

 Innerhalb der einzelnen Help-Texte kennzeichnen verschiedene
 \vSteuerzeichen\v4.  die  Verweise  auf  andere  Help-Texte  sowie
 bestimmte Farbattribute.

 Jeder   Help-Text    kann   eine    beliebige   Anzahl   von
 Kommentarzeilen enthalten,  die jeweils  durch ein Semikolon
 eingeleitet werden.

            \x11\v... über die Help-Engine\v2.\x10  \x11\vDemo-Seite\v5.\x10
;
;
; --  4 : Steuerzeichen innerhalb einer Help-Datei
;
:4
          Steuerzeichen innerhalb einer Help-Datei
─────────────────────────────────────────────────────────────

 Der Help-Compiler erkennt verschiedene Steuerzeichen, die an
 jeder   beliebigen   Stelle   innerhalb   eines   Help-Texts
 angebracht werden können und ihm Nachricht über Verweise und
 Farbattribute   liefern.   Folgende   Steuerzeichen   werden
 erkannt:

  \\v   leitet einen Verweis ein

  \\vN. beendet einen Verweis, wobei N die Nummer des Help-
       Texts angibt, auf den verwiesen wird

  \\u   unterstreicht die nachfolgenden Zeichen  auf  einem
       monochromen Monitor bzw. hebt  sie auf einem Color-
       Monitor farblich besonders hervor

  \\n   stellt die nachfolgenden  Zeichen  wieder in  ihrer
       normalen Farbe dar

  \\    ein normaler Backslash

  \\xhh Zeichen mit dem hexadezimalen ASCII-Code hh

          \x11\v... über die Help-Engine\v2.\x10  \x11\vDemo-Seite\v5.\x10
;
;
; -- 5 : Demo-Seite
;
:5
                         Demo-Seite
─────────────────────────────────────────────────────────────

 Dieser Help-Text  zeigt Ihnen verschiedene Möglichkeiten zur
 Gestaltung eines Help-Texts auf.

            Sie können Text beliebig mit
               Grafik mischen und so
            interessante Effekte erzielen

       Sie können Zeichen besonders \uhervorheben\n.

  Und Sie  können Verweise  anbringen, damit  der Anwender
  auf Wunsch  Hilfe-Informationen über  ein anderes  Thema
  erhalten  kann.  Dieser  \vVerweis\v6.  allerdings  führt  ins
  Leere, nämlich  zum Help-Text  mit  der  Nummer  6,  der
  innerhalb dieser Help-Datei nicht implementiert wurde.

 --->--->--->--->--->--->--->--->--->--->--->--->--->--->--->--->--->--->--->--->---
>--->--->--->--->--->
                        ein Help-Text kann übrigens auch wesentlich breiter als das
Help-Fenster sein

                 \x11\v... über die Help-Engine\v2.\x10
```

Listing 6: *(Ende)*

Schneller, leistungsfähiger und komfortabler:

Das neue QuickC 2.0

Immer mehr Programmierer haben sich in den letzten Jahren von der Leistungsfähigkeit der Programmiersprache C überzeugen lassen und sind dazu übergegangen, PC-Applikationen in dieser Sprache zu entwickeln. Damit ist auch die Nachfrage nach leistungsfähigen und vor allem komfortablen Entwicklungsumgebungen gewachsen, die den Programmierer bei der Programmentwicklung in optimaler Weise unterstützen. Microsoft kommt diesem Trend entgegen, indem es dem Programmierer mit QuickC 2.0 eine Entwicklungsumgebung an die Hand gibt, die keine Wünsche mehr offen läßt.

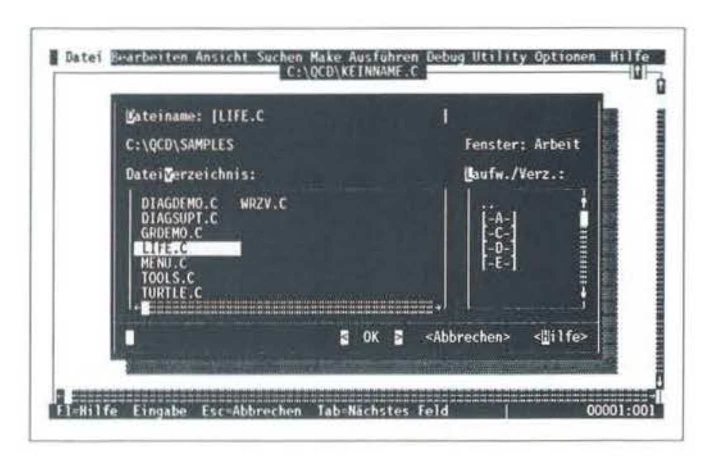


Bild 1: *Die Benutzeroberfläche von QuickC folgt ganz den Richtlinien des SAA-Standards.*

Wer die Programmiersprache C kennt, der weiß, daß diese Sprache hohe Anforderungen an die verwendeten Entwicklungswerkzeuge stellt. Dies beginnt bereits mit dem Befehlssatz, der sich auf ein Mindestmaß an Befehlen zur Programmsteuerung beschränkt und alles Weitere einer sogenannten Library (Funktionsbibliothek) überläßt, die dem Programmierer zahlreiche Funktionen aus den Bereichen Ein-/Ausgabe, Speicherverwaltung, Fließkommaarithmetik etc. zur Verfügung stellt. Da es sich bei C um eine kompilierte Sprache handelt, ist der Programmierer auf einen schnellen Compiler und Linker angewiesen, damit seine Produktivität nicht zu sehr unter langen Übersetzungsläufen leidet. Dabei darf jedoch die Optimierung des Programmcodes nicht zu kurz kommen, denn C bietet dem Compiler durch seine einfache Struktur eine ganze Reihe von Möglichkeiten, den erstellten Maschinencode zu optimieren und dadurch die Ausführungsgeschwindigkeit eines Programms zu steigern. Dies alles sind Leistungsmerkmale, nach denen ein traditioneller C-Compiler üblicherweise beurteilt wird. Doch QuickC ist mehr als nur ein C-Compiler, vereinigt es in sich doch eine komplette Entwicklungsumgebung mit Editor, Make-Utility, Compiler, Linker, Debugger und einem ausgefeilten Hilfe-System.

Die Entwicklungsumgebung von QuickC

Wenn auch die verschiedenen Komponenten wie Compiler und Linker als eigenständige Tools von der DOS-Oberfläche aus aufgerufen werden können, so entfaltet QuickC seine ganze Leistungsfähigkeit doch erst im Rahmen der integrierten Entwicklungsumgebung, in der die verschiedenen Tools unter einem Dach vereinigt und miteinander verknüpft werden. Dem Anwender präsentiert sich dabei eine Benutzerschnittstelle, die ganz nach den Richtlinien des SAA-Standards geformt wurde und mit dem Anwender über Fenster, Pull-Down-Menüs und Dialogboxen in Verbindung tritt (*Bild 1*). Alle Funktionen sind dabei leicht über die Tastatur zu erreichen, doch ist auch die Maus in die Bedienung der Oberfläche voll integriert.

Im Mittelpunkt der Benutzeroberfläche steht das Hauptmenü, über das Sie alle Befehle innerhalb der Entwicklungsumgebung erreichen können und das Dateifenster, in dem jeweils eine Datei editiert wird. Um Anwendern, die bisher mit einem anderen Editor gearbeitet haben, entgegenzukommen, kann der QuickC-Editor beliebig konfiguriert werden. Im Lieferumfang sind dazu vier vordefinierte Tastaturtabellen enthalten, die die 50 Funktionen des Editors auf verschiedene Tastenkombinationen abbilden. Und wer sich mit diesen vordefinierten Tabellen noch immer nicht zurechtfindet, der kann sich mit Hilfe des beigefügten Programms MKKEY eine eigene Tastaturtabelle erstellen.

Aber auch Anwender, die nicht auf ihren gewohnten Editor verzichten möchten, müssen auf den Komfort der QuickC-Entwicklungsumgebung nicht ganz verzichten. Mit Hilfe eines Menüpunktes innerhalb des Optionen-Menüs können neue Menüpunkte in dieses Menü aufgenommen werden. Sie werden mit dem Aufruf eines beliebigen DOS-Programms - z.B. eines Editors - verbunden, so daß der Aufruf des jeweiligen Menüs die Ausführung des damit verbundenen Programms (bzw. DOS-Befehls) zur Folge hat.

Über einen weiteren Menüpunkt innerhalb des Optionen-Menüs wird dem Anwender nun auch die Möglichkeit geboten, zwischen einer Einsteiger- und einer Profi-Version von QuickC umzuschalten. Zwar sind auch in der Einsteiger-Version alle QuickC-Optionen und -Befehle weiterhin vorhanden, doch werden nur noch die wichtigsten von ihnen innerhalb des Hauptmenüs und dessen Untermenüs angezeigt, so daß auch nur sie vom Anwender aufgerufen werden können. Dadurch soll gewährleistet werden, daß sich der Einsteiger bei seiner Arbeit mit QuickC leichter zurechtfindet und erst nach einer Einarbeitungsphase mit allen Möglichkeiten von QuickC konfrontiert wird.

Dem SAA-Standard folgend verfügt QuickC über ein kontextsensitives Hilfe-System, das jederzeit durch Betätigung der Taste [F1] erreichbar ist und auf fast jede Frage eine Antwort hat. Der Begriff »kontextsensitiv« bedeutet in diesem Zusammenhang, daß QuickC bei der



Bild 2: *Das Hilfesystem von QuickC bietet u.a. einen Hilfe-Index an, in dem die Themen aller Hilfeseiten in alphabetischer Reihenfolge aufgeführt werden.*

Suche nach einem geeigneten Hilfstext auf die aktuelle Situation eingeht, aus der heraus der Anwender Hilfe angefordert hat. Wenn Sie sich also z.B. innerhalb des Hauptmenüs oder eines Untermenüs befinden, so wird Ihnen QuickC Hilfe-Informationen zu dem jeweiligen Menüpunkt anbieten und Ihnen so z.B. die Aufgaben dieses Menüpunktes vor Augen führen.

Gleiches gilt auch, wenn Sie sich innerhalb einer der zahlreichen Dialogboxen befinden, die nach dem Aufruf der verschiedenen Befehle aus dem Hauptmenü erscheinen. Auf Anfrage wird Ihnen QuickC dann nicht nur Informationen über die Aufgabe dieser Dialogbox und ihrer einzelnen Felder, sondern auch über die Bedienung, d.h. das Springen zwischen den einzelnen Feldern, die Dateneingabe etc., anbieten.

Etwas anders geht das Hilfesystem allerdings vor, wenn Sie aus dem Dateifenster heraus Hilfe anfordern. Dann nämlich sucht das Hilfesystem nach Informationen im Zusammenhang mit dem Begriff, auf dem sich der Cursor gerade befindet. So können auf diese Art und Weise beispielsweise Informationen über eine der zahlreichen Bibliotheksfunktion eingeholt werden, indem der Cursor auf den Namen dieser Funktion innerhalb eines C-Programms bewegt und dann die Hilfe-Taste betätigt wird. (Alternativ kann auch der Maus-Cursor auf einen bestimmten Begriff bewegt und dann der rechte Mausknopf gedrückt werden.)

Auf dem Bildschirm erscheint augenblicklich eine Hilfeseite, in der alle Informationen über diese Funktion (Aufgabe, Übergabe-Parameter, Return-Wert, zugehörige Include-Datei etc.) aufgeführt werden. Zusätzlich hält das Hilfesystem zu jeder Funktion aus der Laufzeit-Bibliothek auch ein kleines Demo-Programm bereit, das den Einsatz dieser Funktion im Rahmen eines Programms verdeutlicht. Wie auch jeder andere Text innerhalb einer Hilfe-Seite, so können auch Teile dieses Listings direkt in Ihr Programm kopiert werden. In bisherigen QuickC-Versionen mußten viele dieser Informationen mühsam in einem speziellen Verzeichnis der Funktionen aus der Laufzeitbibliothek nachgeschlagen werden. Dies ist nun nicht mehr notwendig, da alle diese Informationen im elektronischen Hilfe-System zu finden sind, und man hat sich bei Microsoft daher sogar entschlossen, die gedruckte »Referenz der Laufzeitbibliothek« nicht mehr mit QuickC auszuliefern.

Neben kontextsensitiven Informationen kann man über das Hilfesystem aber auch an allgemeine Informationen gelangen. Zu diesem Zweck gibt es im Hauptmenü einen speziellen Menüpunkt, der auf allgemeine Themen rund um QuickC und die C-Programmierung verweist, sowie einen Hilfe-Index (*Bild 2*), in dem die Themen der Hilfeseiten in alphabetischer Reihenfolge aufgeführt werden.

Der QuickC-Compiler

Der QuickC-Compiler ist in der Version 2.0 voll kompatibel zu Microsofts großem C-Compiler (MSC) der Version 5.1, kann im Gegensatz zu diesem aber noch nicht im Protected-Modus von OS/2 betrieben werden. Dafür unterstützt er aber die neuen Funktions-Deklaratoren, die mit der Version 5.1 eingeführt wurden. Im einzelnen sind dies `_export` (für die Windows-Programmierung), sowie `_saveregs` und `_loadds`, die hauptsächlich bei der Programmierung von Interrupt-Handlern in C zum Einsatz kommen.

Durch interne Optimierungen und den Einsatz einer neuen Technik, des sogenannten »inkrementalen Kompilierens« konnte die Kompilierungsgeschwindigkeit des QuickC-Compilers gegenüber seinen Vorgängerversionen noch einmal deutlich gesteigert werden. Dabei verbirgt sich hinter dem inkrementalen Kompilieren eine recht einfache Idee, die auf die Art und Weise vorgeht, wie die meisten Programmierer ihre Programme entwickeln.

In der Regel vollzieht sich diese Entwicklung nämlich Schritt für Schritt (bzw. Funktion für Funktion), wobei der Programmierer das Programm nach der Erstellung jeweils einer logischen Einheit von Funktionen übersetzen läßt und die einwandfreie Ausführung der Funktionen dann austestet. Dadurch aber muß in der Regel nicht das komplette Programm neu übersetzt werden, denn viele Funktionen wurden seit der letzten Kompilierung nicht verändert und resultieren deshalb auch nicht in neuem Maschinencode.

Dieser Idee folgend kompiliert QuickC bei jedem Übersetzungslauf tatsächlich nur die Funktionen, an denen seit der letzten Kompilierung Änderungen vorgenommen worden sind. Um welche Funktionen es sich dabei handelt, erkennt der Compiler anhand einer sogenannten »Module Description Table« (einer Datei mit der Erweiterung `MDT`), die er bei jeder Kompilierung einer Quelldatei anlegt bzw. ergänzt. Eine Ausnahme bilden dabei lediglich Änderungen an den globalen Variablen, die in der Regel Veränderungen in fast allen Funktionen nach sich ziehen. Da der Compiler diese Veränderungen nicht so ohne weiteres nachvollziehen kann, wird in diesem Fall die komplette Quelldatei neu kompiliert.

```
void Cls( unsigned int VioSeg, unsigned char Farbe )
{

 /*-- hier können beliebige C-Befehle stehen ---------------*/

 _asm                        /* es folgt ein Assembler-Block */
  {
   mov  bx,di                ;DI darf nicht verändert werden
   mov  es,VioSeg            ;übergebene Segmentadresse nach ES
   mov  ah,Farbe             ;Farbe nach AH
   mov  al,' '               ;ASCII-Code
   mov  cx,2000              ;d. Bildschirm umfaßt 2000 Zeichen
   xor  di,di                ;am Anfang des Video-RAM beginnen
   rep  stosw                ;den Video-RAM mit AX füllen
   mov  di,bx                ;Inhalt von DI zurückholen
  }

 /*-- hier können beliebige C-Befehle stehen ---------------*/

}

void main( void )
{
 Cls( 0xb000, 0x70 );       /* monochromen Video-RAM löschen */
}
```

Listing 1: *Dieses Programm verdeutlicht die Einbindung von Assembler-Befehlen in ein C-Programm.*

Treten bei der Kompilierung Fehler auf, öffnet QuickC auf dem Bildschirm ein spezielles Fenster, in dem alle Fehlermeldungen des Compilers aufgeführt werden. Während man mit Hilfe einfacher Tastenkombinationen von Fehler zu Fehler springt, zeigt QuickC im Dateifenster automatisch die Zeile innerhalb der Quelldatei an, in der der Fehler aufgetreten ist, so daß man den Fehler direkt korrigieren kann und sich nicht mühsam durch die Datei »blättern« muß.

Assembler-Befehle in C-Programmen

Eine weitere Neuheit stellt die Fähigkeit des QuickC-Compilers dar, Assembler-Befehle innerhalb eines C-Programms zu assemblieren und direkt in den C-Code zu integrieren (*Listing 1*). Über das neue Befehlswort _asm können dabei beliebige Assembler-Sequenzen in den Quellcode aufgenommen und während der Kompilierung direkt in Maschinensprache übersetzt werden. Diese Vorgehensweise empfiehlt sich besonders bei kleineren Assembler-Routinen bzw. -Sequenzen, da dadurch nicht auf einen speziellen Assembler zurückgegriffen und die QuickC-Umgebung nicht verlassen werden muß.

Der QuickC-Linker

Auch im Hinblick auf den Linklauf und den ihm zugrunde liegenden Linker hat sich gegenüber der Vorgängerversion einiges verändert. QuickC verfügt nun über einen eigenen Linker, der auf die inkrementale Kompilierung eingeht und deshalb im Englischen als »incremental linker« bezeichnet wird. Darüber hinaus wird der Linklauf zwar weiterhin von der QuickC-Oberfläche aus gestartet, doch wurde von dem bisherigen Konzept der »Quick-Library« abgewichen. Dadurch können Programme nun auch innerhalb der QuickC-Umgebung unter allen fünf Speichermodellen kompiliert und auf alle Funktionen der Laufzeit-Bibliothek zurückgegriffen werden, ohne daß dafür eine spezielle Quick-Library erstellt werden muß. Der Preis dafür ist eine geringere Geschwindigkeit beim Linklauf, doch wird dies durch die Geschwindigkeitssteigerungen beim Kompilieren mehr als wett gemacht.

Die Laufzeit-Bibliothek selbst baut auf der Bibliothek des großen Microsoft C-Compilers auf und verfügt dementsprechend neben den Standard-Funktionen (zur Ein-/Ausgabe, Speicherverwaltung und Fließkommaarithmetik) auch über spezielle »PC-Funktionen« (zum Zugriff auf das DOS und BIOS) sowie einen Satz leistungsfähiger Grafikfunktionen, die alle bekannten Video-Karten von MDA über CGA und die Hercules Grafikkarte bis hin zu EGA, VGA und MCGA unterstützen.

Die Laufzeit-Bibliothek des QuickC-Compilers begnügt sich damit jedoch nicht, sondern erweitert den Bereich der Grafikfunktionen noch um eine Reihe von Funktionen zur Erstellung von Präsentationsgrafiken. Mit ihrer Hilfe kann der Anwender Balken-, Kuchen-, Linien- und Punktdiagramme erzeugen, ohne sich um die Skalierung der Axen, die Zentrierung von Titeln etc. sorgen zu müssen. Mit nur wenigen Programmzeilen lassen sich dadurch Grafiken programmieren, die bisher nur mit speziell dafür konzipierten Programmen erstellt werden konnten.

Bei der Erstellung derartiger Grafiken greift QuickC u.a. auf mehrere Schriftdateien zurück, mit deren Hilfe Zeichen in fast beliebiger Größe auf den Bildschirm gebracht werden können. Neben den Schriftmustern »Courier«, »Helvetica« und »Times Roman«, die als Bitmuster vorliegen und daher nur in drei Schriftgrößen ausgegeben werden können, sind im Lieferumfang von QuickC auch Schriftdateien für die Schriftmuster »Modern«, »Script« und »Roman« enthalten, die vektororientiert aufgebaut sind und daher in jeder beliebigen Größe auf den Bildschirm gebracht werden können.

Die Make-Utility

Der Leistungsumfang von QuickC 2.0 zeigt, daß dieses Produkt nicht nur für C-Einsteiger, sondern auch für fortgeschrittene, wenn nicht gar professionelle Programmierer entwickelt wurde. Die aber erstellen häufig große Programme, die der besseren Übersichtlichkeit halber in einzelne Module aufgeteilt werden (*Bild 3*). Zwar werden diese Module separat kompiliert, doch beim Linklauf zu einem einzigen Programm zusammengefügt. Wichtig ist dabei, daß alle Module auf dem »neuesten Stand« sind und gerade hier schleichen sich sehr leicht Fehler ein.

Wenn beispielsweise mehrere Module auf eine Include-Datei zurückgreifen, dann müssen alle diese Module neu kompiliert werden, sobald Veränderungen in dieser Datei vorgenommen werden, weil diese Veränderungen möglicherweise die Codegenerierung durch den Compiler beein-

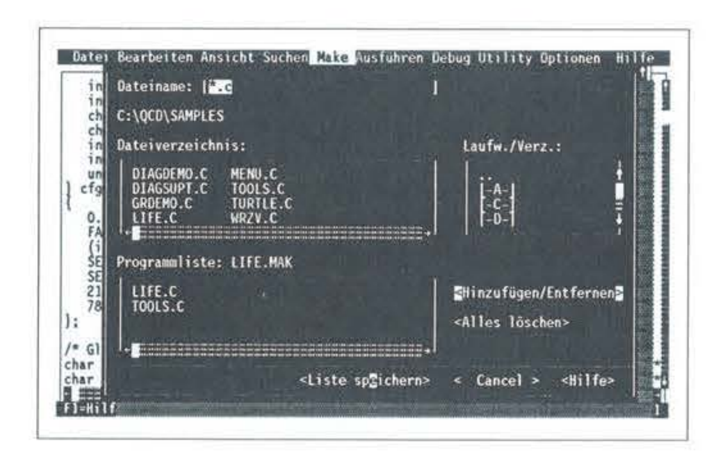


Bild 3: *Auf dem Bildschirm kann der Anwender mit Hilfe der integrierten Make-Utility die Module definieren, aus denen sich ein Programm zusammensetzt.*



Bild 4: *Hier wird während der Programmausführung der Inhalt verschiedener Variablen permanent am Bildschirm angezeigt.*

flussen (z.B. wenn eine Konstante verändert wird, auf die innerhalb der verschiedenen Module zugegriffen wird).

Hier behilft man sich häufig, indem man Abhängigkeiten zwischen verschiedenen Dateien definiert und so z.B. festlegt, daß die C-Module `x`, `y` und `z` neu kompiliert werden müssen, sobald Veränderungen an der Include-Datei `h` vorgenommen werden. Während diese Abhängigkeiten normalerweise in einer relativ komplizierten Syntax beschrieben und in einer speziellen Make-Datei zusammengefaßt werden müssen, macht es QuickC dem Anwender hier viel leichter.

Über einen speziellen Menüpunkt im Hauptmenü erlaubt QuickC dem Anwender die Erstellung einer solchen Make-Datei und gleichzeitig die Definition aller Dateien, die zur Erstellung des jeweiligen Programms von Bedeutung sind. Will man dieses Programm erstellen, stellt QuickC dann automatisch fest, ob und welche Dateien dazu neu kompiliert werden müssen und startet anschließend automatisch den Linklauf, an dessen Ende das ausführbare Programm steht.

Der Debugger

Ob man mit QuickC nun große oder kleine Programme erstellt - nur selten funktionieren diese Programme auf Anhieb einwandfrei und so muß man sich häufig auf die Fehlersuche (Debugging) machen. Die Zeiten, in denen man sich dabei mit der Ausgabe bestimmter Informationen während des Programmablaufs über `printf`-Befehle begnügen mußte, sind allerdings längst vorbei und so verfügt auch QuickC über einen integrierten Debugger, der die Fehlersuche wesentlich erleichtert.

Er erlaubt die Verfolgung des Programmablaufs auf dem Bildschirm, wobei jeweils die aktuell ausgeführte Quellzeile im Dateifenster hervorgehoben wird. Nach der Ausführung jeder Quellzeile kann der Anwender die Programmausführung stoppen, den Inhalt beliebiger Variablen betrachten und sogar verändern, um dadurch auf die Programmausführung Einfluß zu nehmen (*Bild 4*). Wem das nicht ausreicht, der kann sich am Bildschirm auch gleich die Inhalte der verschiedenen Prozessorregister anzeigen lassen und so nachvollziehen, wie die einzelnen C-Befehle den Inhalt dieser Register verändern.

Auch das Setzen von Haltepunkten ist möglich, so daß die Programmausführung ungestört fortgesetzt wird, bis einer der gesetzten Haltepunkte erreicht wird. Besonders schwierig zu entdeckende Fehler, wie etwa der Speicherzugriff mit einem nicht initialisierten Zeiger, lassen sich entdecken, indem eine bestimmte Bedingung (beispielsweise `*ptr == 5`) definiert wird, bei deren Eintritt die Programmausführung gestoppt wird.

Wem auch diese Möglichkeiten noch zu begrenzt erscheinen, der kann zum Debuggen übrigens jederzeit auf CodeView, den Debugger der großen Microsoft-Compiler, umsteigen, denn die von QuickC erstellten Dateien können von CodeView ohne weiteres verarbeitet werden.

Fazit

Der QuickC-Compiler erweist sich in der täglichen Praxis als zuverlässiges und komfortables Entwicklungswerkzeug, das auch zur Realisation großer Projekte herangezogen werden kann. Dies wurde auch bei der Erstellung unserer SAA-Serie deutlich, die mittlerweile einen Quellcode von mehr als 300 Kbyte umfaßt und komplett mit QuickC entwickelt wurde. Lediglich der Debugger sollte meiner Meinung nach noch um die Möglichkeit erweitert werden, auch den erzeugten Maschinencode anzuzeigen, da dies manchmal sehr hilfreich sein kann. Vor allem, wenn man Code formuliert, den der Compiler zwar richtig, aber dennoch anders interpretiert, als man es selbst getan hätte, erweist sich dies bei der Fehlersuche oft als der einzige Weg zum Erfolg.

Michael Tischer

Carbon Copy Plus, Version 5.1:

Der ferngesteuerte Computer

Ein verzweifelter PC-Benutzer ruft an und fragt um Rat, weil sein Programm plötzlich nicht mehr macht, was es soll. Ein Programm muß neu konfiguriert werden, aber dazu müßte man erst zum Standort der Installation fahren. Auf einem PC hat jemand versehentlich eine Programmdatei gelöscht, weiß aber nicht genau, welche. Leider sitzt der unglückliche Benutzer 300 km entfernt. - Nun sitzt der Experte am Telefon und versucht mit viel Mühe, dem oft unerfahrenen Benutzer komplizierte DOS-Kommandos zu diktieren. Wenn man nur per Telefon in den weit entfernten PC hineinschauen könnte!

Man kann. Eine besondere Art von Kommunikationsprogrammen macht es möglich, einen PC über eine Telefonleitung fernzusteuern. Die bekanntesten Programme dieser Art sind *Remote* (1979), *Carbon Copy*, *Close-Up* (1985), *PC Anywhere* (1986) und *Co/Session* (1987). Remote existiert mittlerweile in den neuen Versionen Remote2 und R2LAN. Wir haben die ebenfalls neue Version 5.1 von *Carbon Copy Plus* getestet.

Beim Fernsteuern werden alle Tastendrücke auf den anderen PC übertragen und alle Bildschirmveränderungen von diesem zurückgemeldet, so daß der »Helfer« auf seinem Bildschirm eigentlich den Bildschirm des anderen Computers sieht. Zum Übertragen braucht man an jedem der beiden PCs ein Modem. Wenn man will, kann man Übertragungsfehler automatisch korrigieren lassen. Die Fehlerkorrektur verlangsamt allerdings den Prozeß.

Auf der »Host«-Seite, d.h., auf dem zu beobachtenden Computer, muß zuerst ein residentes Programm (CC.EXE, 52448 Byte) geladen werden. Danach kann der Benutzer beliebige Programme ablaufen lassen, sogar Terminal-Emulationsprogramme wie IRMA 3270 oder IBM 5250. Diese Programme können dann vom zweiten PC, dem »Helfer«, ferngesteuert werden.

Auf dem »Helfer«-PC läuft das Programm CCHELP.EXE, das zwar mehr Speicherplatz belegt, sich diesen aber nicht mit einem anderen Programm teilen muß. CCHELP dient nur zur Fernsteuerung des anderen Computers über die Telefonleitung.

Grafische Bildschirme

Carbon Copy geht sogar so weit, Grafikbildschirme zu übertragen. Dabei wird selbst dann der Bildschirminhalt der Gegenstelle angezeigt, wenn diese einen ganz anderen Grafikadapter mit einer anderen Auflösung verwendet. Die Umsetzung kann zwangsläufig nicht ganz perfekt sein, besonders wenn der empfangende PC eine sehr niedrige Auflösung hat, aber für viele Zwecke reicht das aus.

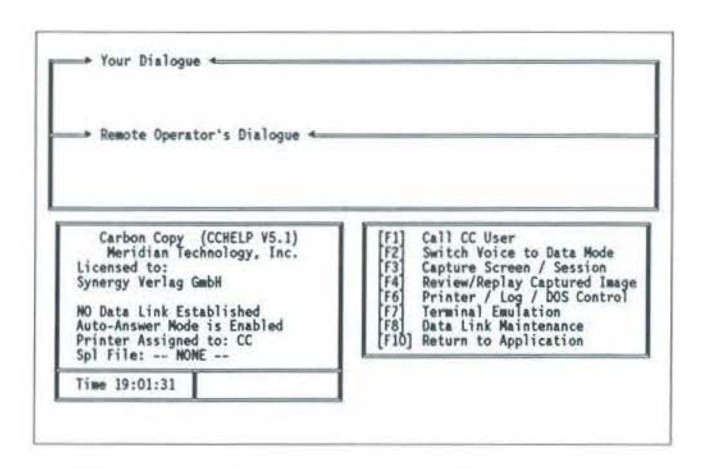


Bild 1: *Nach dem Start des »Helfer«-Programms CCHELP zeigt sich ein übersichtlich gestalteter Bildschirm mit Hinweisen zur Funktionstasten-Bedienung.*

Allerdings darf man sich keiner Täuschung darüber hingeben, wie langsam eine solche Übertragung vor sich geht. Mit 1200-Baud-Modems ist es bei höheren Auflösungen fast unerträglich und mit 300 Baud sollte man es besser gar nicht erst versuchen. Mit einem 9600-Baud-Modem ist die Geschwindigkeit immer noch lange nicht so, als ob man direkt am anderen Computer säße. Das ist selbst bei 38,4 Kilobaud, der höchsten unterstützten Geschwindigkeit, noch nicht der Fall, aber für den Zweck einer längeren Fernwartung sind 9600 Baud sicher sehr erstrebenswert. Man merkt hier, daß der Hersteller Meridian Technology eine Tochtergesellschaft des Modemherstellers Microcom ist, der besonders durch sein fehlerkorrigierendes MNP-Protokoll bekannt wurde. MNP-Modems beschleunigen die Datenübertragung durch Datenkompression, was dem Programm Carbon Copy sicher sehr zugute kommt.

Dateiübertragung mit »CCDOS«

Was tun, wenn eine Datei fehlt, die man aber schon auf dem »Helfer«-Computer hat? Kein Problem, mit Carbon Copy kann man auch beliebige Dateien über die Leitung schicken. Dazu bietet das Programm einen Arbeitsmodus, der den DOS-Prompt simuliert. Dieser Modus heißt »CCDOS« und erlaubt die Kommandos ALERT, CHDIR, COPY, DEL, DIR, EXIT, MKDIR, PROMPT, REN und TYPE. ALERT, das einzige dieser Kommandos, das es in DOS nicht gibt, löst einen Warnton aus. Die übrigen funktionieren genau wie die gleichnamigen DOS-Kommandos. Damit man zwischen dem lokalen Laufwerk C (oder A, B, usw.) und dem gleichnamigen Laufwerk C auf dem »Host«-Computer unterscheiden kann, werden die Laufwerke mit LC (lokales C) und HC (Host-C) bezeichnet. Mit dem Kommando

```
COPY LC:123.CNF HC:
```

kann man also die Datei 123.CNF vom eigenen PC auf den anderen kopieren. Carbon Copy sendet oder empfängt

die gewünschte Datei mit Fehlerkorrektur über die Telefonleitung. Man kann Dateien sogar übertragen, während der Benutzer auf der Host-Seite mit einem beliebigen anderen Programm arbeitet, weil Carbon Copy die Dateien im Hintergrund überträgt.

Verständigung mit dem Bediener

Während einer Verbindung kann man jederzeit die Bildschirmanzeige des gerade untersuchten Programms verlassen und auf ein Carbon-Copy-Menü umschalten. Dort findet man auch zwei »Chat Windows« (frei übersetzt: Plauder-Fenster), in denen man sich mit dem Bediener der Gegenseite in beiden Richtungen per Tastatur verständigen kann. Bestimmte Modems können auch von »Daten« auf »Sprache« umgeschaltet werden, so daß man zwischendurch auch mündlich ein paar Worte wechseln kann. Wenn solche Modems verwendet werden, dann stellt Carbon Copy eine entsprechende Funktionstaste zum Umschalten von Daten auf Sprache und umgekehrt bereit.

Besondere Beachtung findet das Drucken. Carbon Copy kann den Druck zum »Helfer«-Computer umleiten, so daß der Helfer auch den Ausdruck direkt prüfen kann. Dabei werden Spool-Möglichkeiten bereitgestellt, so daß man nicht auf den Drucker warten muß. Carbon Copy kann auch auf beiden Computern gleichzeitig drucken.

Besondere Eigenschaften

Wer ein Programm wie Carbon Copy lädt und sein Modem auf automatische Anrufbeantwortung stellt, der muß damit rechnen, daß irgendwann ein Hacker versucht, in seinen Computer einzudringen. Zum Schutz dagegen bietet Carbon Copy die Zugangskontrolle durch Paßwort, den automatischen Rückruf und die Verschlüsselung aller übertragenen Daten. Das Verschlüsseln sichert die Daten nicht nur vor Eindringlingen, sondern auch vor dem Abhören an beliebiger Stelle im Telefonnetz.

Carbon Copy enthält einen Leckerbissen für Programmierer. Es ist möglich, einige Funktionen auch von einem anderen Programm aus zu steuern oder abzufragen, so daß man in bestimmten Programmen die Zusammenarbeit mit Carbon Copy schon vorsehen und vorprogrammieren kann.

Außer der komplizierten Fernsteuerung anderer Programme kann man mit Carbon Copy auch einfache, normale Kommunikationsaufgaben lösen. Für eine solche Terminal-Emulation entlehnt das Programm die meisten wichtigen Kommandos aus dem bekannten Kommunikationsprogramm *Crosstalk XVI*. Sogar die Script-Sprache wird unterstützt. Wie bei Crosstalk XVI reicht jedoch auch diese nicht bis zu Mehrfachverzweigungen in Abhängigkeit von verschiedenen empfangenen Daten. Carbon Copy emuliert eine Reihe populärer Terminals und unterstützt wie Crosstalk die File-Transfer-Protokolle Xmodem/Batch, Ymodem/Batch sowie Kermit im Text- und Binär-Modus.

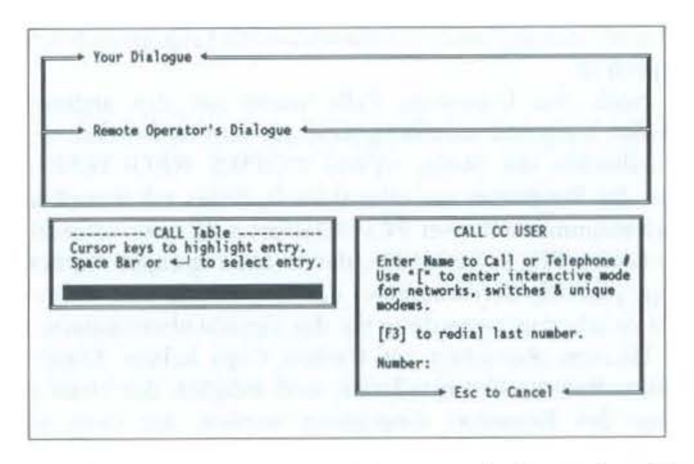


Bild 2: *Nach Betätigen der Funktionstaste* [F1] *verändert sich der untere Bereich des Bildschirms zum Eingeben oder Auswählen der gewünschten Telefonnummer.*

Eine »Schnappschuß«-Taste (Alt-W) ermöglicht es, jederzeit den Bildschirminhalt in eine Datei zu speichern, z.B. um ihn später zu analysieren oder vorzuführen.

Probleme

Das beste Programm nützt nichts, wenn es nicht geladen ist. Wenn man Ferndiagnosen unerwartet auftretender Fehler mit Carbon Copy durchführen will, dann muß man dafür sorgen, daß das Programm immer »auf Verdacht« geladen ist.

Ein Problem kann auftreten, wenn das ferngesteuerte Programm selbst versucht, den verwendeten Kommunikationsanschluß zu benutzen oder zurückzusetzen. Das Handbuch warnt z.B. ausdrücklich vor Lotus Symphony. Beim Test zeigte sich jedoch, daß Carbon Copy den COM-Anschluß sehr wirksam vor dem Zugriff anderer Programme schützt. Es hatte mit Symphony trotz installiertem COM-Treiber keine Probleme. Der DOS-Befehl MODE zeigte an, daß der COM-Anschluß gar nicht mehr vorhanden wäre. Natürlich kann das ferngesteuerte Programm den Anschluß dann auch nicht benutzen.

Programme, die einen eigenen Tastaturtreiber verwenden, können möglicherweise nicht mit Carbon Copy ferngesteuert werden. Wir waren nicht in der Lage, Microsoft Windows mit Carbon Copy Plus (Foreign Keyboard Version) fernzusteuern. Allerdings wäre es wegen seiner grafischen Arbeitsweise und der damit verbundenen großen Bildschirm-Datenmengen auch etwas vermessen, dieses über eine Telefonleitung bedienen zu wollen.

Da Carbon Copy auf der Host-Seite immer als residentes Programm geladen werden muß, sind die üblichen Konflikte mit anderen, konkurrierenden TSR-Routinen oder mit anderweitig empfindlichen Programmen zu erwarten. Bei unseren Tests mit einer kleinen Auswahl verschiedener Programme traten keine Schwierigkeiten auf. Man sollte sich jedoch vor dem Kauf davon überzeugen, daß Carbon

Copy mit den geplanten fernzusteuernden Programmen verträglich ist.

Noch eine besondere Falle wartet auf den arglosen Käufer. Klein und unauffällig steht auf dem Buchrücken des Handbuches die Notiz: »TWO COPIES REQUIRED«. Wer das Programm nur einmal kauft, dieses mit derselben Seriennummer auf zwei PCs installiert und dann versucht, die beiden PCs zu verbinden, der ist hineingetappt. Carbon Copy prüft die Seriennummer der Gegenstelle und weigert sich zu arbeiten, wenn diese mit der eigenen übereinstimmt.

Hiervon abgesehen hat Carbon Copy keinen Kopierschutz. Bei der Erstinstallation muß lediglich der Firmenname des Benutzers eingegeben werden, der dann im Lotus-Stil in das Programm »eingebrannt« wird. Er läßt sich nie wieder entfernen und wird bei jedem Programmstart angezeigt. Die Lizenz untersagt nur die gleichzeitige Benutzung des Programms auf mehreren Computern.

Das Handbuch ist gut gegliedert und vermittelt in knappem Stil die notwendigen Informationen, allerdings, ebenso wie das Programm selber, in englischer Sprache. Insgesamt macht das Programm Carbon Copy Plus einen soliden Eindruck. Besonders die leistungsfähige Behandlung von Grafikbildschirmen verdient Anerkennung.

Weitere Einsatzmöglichkeiten

Die Einsatzmöglichkeiten eines Programms wie Carbon Copy gehen weit über die hier geschilderte Fernwartung, Fernbetreuung und Ferndiagnose hinaus. Carbon Copy kann z.B. als eine preiswerte Alternative zum Anschluß einer Terminal-Emulation an einen Großrechner verwendet werden, besonders wenn diese nur gelegentlich gebraucht wird. Dazu schließt man vor Ort an den Großrechner einen PC mit Terminal-Emulation an, lädt auf diesem aber gleichzeitig das Carbon Copy Host-Programm. Dann können mehrere andere PCs reihum über das Telefonnetz auf diesen zugreifen und auf diese Weise am Großrechner arbeiten, ohne selbst einen teuren Anschluß zu benötigen.

Andere Möglichkeiten beruhen darauf, Carbon Copy auf einem PC ablaufen zu lassen, der in ein Local Area Network (LAN) eingebunden ist. Dadurch haben andere, räumlich entfernte PCs die Möglichkeit, auf die Ressourcen des LAN und seiner Server, z.B. auch Kommunikationsserver, zuzugreifen, ohne über teure Brücken oder Gateways an dieses angeschlossen sein zu müssen. Ein weiterer Vorteil dieser Arbeitsweise ist, daß auf diesem Wege oft nicht ganze Dateien gelesen und über das LAN übertragen werden müssen, sondern nur die Informationen der Bildschirme, die der Fern-Benutzer tatsächlich sehen will. Dies kann erheblich rationeller sein als ein LAN-Anschluß. Sicher werden in der nächsten Zukunft noch viele weitere interessante und nützliche Verwendungen für den »ferngesteuerten PC« gefunden werden.

Hans-Georg Michna

Impressum

Das *Microsoft System Journal* erscheint alle zwei Monate (ungerade Monatszahlen) etwa Mitte des Vormonats.

Herausgeber, verantwortlich und Anschrift der Redaktion:
Microsoft GmbH, Redaktion *Microsoft System Journal*,
Edisonstr. 1, D-8044 Unterschleißheim
Telefon: 089 / 31705-0, Telex: (17) 898 328, Telefax: 089 / 31705-400

Redaktion:
Günter Jürgensmeier, Haar und Hartmut Niemeier, Wildenberg

Mitarbeiter dieser Ausgabe:
Marcellus Buchheit, Dominique Guignol, Rainer G. Haselier, Ed Iacobucci, Günter Jürgensmeier, Hans-Georg Michna, Hartmut Niemeier, Michael Tischer

Manuskripteinsendungen:
Manuskripte und Programmlistings werden von der Redaktion gerne angenommen. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck und zur Vervielfältigung der Programmlistings auf Datenträgern. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen. Nicht zur Veröffentlichung gelangte Manuskripte und Listings können nur zurückgeschickt werden, wenn Rückporto beiliegt.

Titelgestaltung: Hermann Menig
Anzeigenverkauf: Marianne Nuß
Druck: Alois Erdl KG, Trostberg
Abonnements: Microsoft GmbH
Bezugspreise: Das Einzelheft kostet DM 19,80. Der Abonnementpreis beträgt DM 115,- für 6 Ausgaben und DM 210,- für 12 Ausgaben. Zu den einzelnen Ausgaben ist zum Preis von DM 19,80 eine Diskette mit allen Listings erhältlich. Das Abonnement inklusive Diskette kostet DM 230,- für 6 Ausgaben und DM 420,- für 12 Ausgaben. In den Preisen enthalten sind Mehrwertsteuer, Versandkosten und Zustellgebühren. Auslandsbezug auf Anfrage. Sollte die Zeitschrift aus Gründen, die nicht vom Herausgeber zu vertreten sind, nicht geliefert werden können, besteht kein Anspruch auf Nachlieferung oder Erstattung vorausbezahlter Bezugsgelder.
Bezugsmöglichkeiten: In Buchhandlungen und im Computer-Fachhandel. Abonnements und Einzelbestellungen: Microsoft GmbH.

Das *Microsoft System Journal* wird mit Microsoft Word 4.0 geschrieben und gestaltet. Der Ausdruck erfolgt mit den HP-Softfonts AD und AF und dem Programm- und Schriftenpaket *DocuJet* auf einem HP-LaserJet Series II.